(1)

FINAL TECHNICAL REPORT
VOLUME I

INTERACTIVE VISUAL SIMULATION OF COMMUNICATION SYSTEMS

April 29, 1988

Contract No. DAAB07-87-C-A023

Prepared for

Commander

U.S. Army CECOM

Fort Monmouth, N. J. 07703-5000

LINKNET

710 Silver Spur Road, Suite 285

Rolling Hills Estates, California 90274

(213) 373-3384

TABLE OF CONTENTS

LIST OF FIGURES

1. EXECUTIVE SUMMARY

This report describes the Phase I SBIR effort towards developing a visual communications link and network simulation system operating on a Definicon DSI-750 68020 co-processor board enhanced IBM PC/XT-compatible computer. This work was performed under Contract No. DAAB07-87-C-A023 to the U.S. Army Communications Electronics Command. The report is organized into three volumes. This first volume describes the architecture and capabilities of the visual simulation software, the conclusions arrived at during this effort, and recommendations for further Phase II work. The first volume, which contains a detailed description of the operation of the developed software, also serves as the user manual. Volume I is organized into six sections. Section 2 describes the overall software architecture. The user interface software and user operation procedures are described in Section 3. Sections 4 and 5 describe the capabilities of the link simulation library and the network simulation library respectively. Section 6 describes the graphics display capabilities. Finally, conclusions and recommendations for further work are given in Section 7.

Appendix A of Volume I describes the configuration and installation procedure for the visual simulation software on the Definicon DSI-750 co-processor board enhanced Compaq Deskpro computer delivered to U.S. Army CECOM. Appendix B describes the disk files used for storing simulation results. Volume II contains the source code listings of the link and network simulation software while the user interface software source code is given in Volume III.

2. SOFTWARE SYSTEM ARCHITECTURE

A primary objective in this effort was to develop a communications link and network simulation software system that does not require programming by the user. Thus, a "user dialogue" approach was adopted in which the user configures the system to be simulated via a dialogue with the simulation software. Such a system contains the following four major modules:

* model library
* system configurator
* simulation operating system
* postprocessor

The model library contains the functional blocks which simulate specific operations in a communications link or network. Sections 4 and 5 below describe the capabilities of the link and network simulation libraries respectively. The system configurator is used to select a set of functional blocks from the model library and connect them in the topology specified by the block diagram of the system being simulated. In the dialogue approach, a user-friendly interface to the system configurator is necessary. The approach taken in this effort towards the design of a user-friendly interface was to incorporate the use of mouse-driven user inputs with "Lotus Symphony-like" slash-bar menus. Visual displays are provided wherever feasible. The interface software, described in Section 3 below, performs the tasks of both the user-interface and the system configurator. The execution of the simulation run and management of the model library is governed by the simulation operating system. The interface software allows the user to control the simulation run through this operating system. The simulation operating system is described in detail below. Finally, the postprocessor accepts time histories of simulation data such as signal waveforms, bit-error statistics, packet delays and throughputs and displays the results to the user. Both data and graphics displays are provided. Animated displays during the simulation run as well as post-run displays of dynamically evolving statistics are included. The post-processor function is shared by the functional blocks and dedicated executable graphics generating routines. The user controls the active displays through the interface. Discussion of the post-processing graphics capabilities is given in Section 6 along with related descriptions in the following section on the interface software, and in Sections 4 and 5 on the model libraries.

2.1 Link Simulation Operating System

The communications link model library must include various functional blocks such as signal sources, modulators and demodulators, encoders and decoders, transmitter and receiver filters, and transmission channels. The baseline link library developed in this effort consists of 45 separate functional blocks. The link simulation operating system must have the

capability of effectively managing the availability of this large library to the system configurator and to the simulation exerciser executing the simulation run. In addition, it should provide a common framework for the development of the large number of functional blocks in the model library. In this effort, we have chosen to develop the simulation operating system based on BLOSIM (an acronym for BLOck SIMulator), which is a general purpose C language simulation programming environment developed by D.G. Messerschmidt [1]. We have acquired the right of modifying and using BLOSIM for our simulation operating system.

BLOSIM primarily provides a time-driven simulation environment for developing and running C language programs for simulating sampled-data systems. Any system to be simulated must be divided into an interconnection of functional blocks. Users of BLOSIM must program functional block modules in C programming language. In the BLOSIM environment, blocks exchange data using intermediate first-in first-out (fifo) buffers. BLOSIM sets up and manages these fifo's. In order to use BLOSIM to simulate a system, the user must first program the functional blocks required in the system to be simulated and compile these functional block source code modules with a C compiler to obtain corresponding object code modules. The user must also specify the system topology that describes the blocks in the system and the interconnections between them. The system topology is specified in a C program referred to as the "UNIVERSE" in BLOSIM terminology. BLOSIM consists of three run-time object code modules: FIFO, BLOCK and SEQUENCER. FIFO contains the routines which perform the creation and management of the fifo's during the simulation run. BLOCK routines keep track of and manage the blocks (called STAR's in BLOSIM terminology) specified in the UNIVERSE. The function of SEQUENCER is to schedule and control the order of execution of the blocks during the simulation run. The blocks are scheduled and executed in the order of "signal flow" as specified by the system interconnection topology. Prior to the start of the simulation, the universe source code is compiled and linked with the object code modules of the functional blocks in the simulated system and with the BLOSIM run-time object modules to produce the final executable code for the simulation run.

There are considerable advantages to using BLOSIM for developing the link simulation model library. First block routines can have different parameters declared at run time, and thus be used several times in a system and also interchangeably in different systems. The division of the system into identifiable and functionally contained blocks results in a desirable modularity. The software implementing the blocks is written to a standard fifo interface, and do not interact with other blocks directly. This hides the internal implementation of any particular block from other blocks, and insures that blocks can be interconnected effortlessly even when the blocks are written by different programmers. This capability greatly enhances the maintainability of the simulation software.

The link simulation architecture is as follows. The link portion of the model library consists of the object code modules of the functional blocks used in link simulations. The interface software performs the task of the system configurator. It generates the UNIVERSE source code from the user system configuration specifications. The interface software also initiates a simulation by performing the compilation of the UNIVERSE and the linking to the library object code modules and the BLOSIM run-time object modules to produce the executable simulation code. It then starts the simulation run by loading the executable code. The execution of the simulation run is governed by the BLOSIM routines linked in the executable code. Post-run displays are accessed through the interface software, which loads the appropriate executable graphics generating routines.

2.2 Network Simulation Operating System

The baseline communications network simulation model library developed in this effort has the capability of simulating certain store-and-forward packet communication networks and multiple-access packet communications networks. The operation of a packet communications network is analogous to the operation of a distributed system in which a collection of parallel processes evolve with interaction between the processes. The processes are the states of the nodes in the network, and the interaction between the nodes involves the sending of packets between nodes. The appropriate method for simulating such systems is by an event-driven approach. In event-driven simulations, the state of the system being simulated changes according to the occurrence of a sequence of events. Event-driven simulation programs are centered on a sequence of pairs (e_i,t_i), where e_i is the event that occurs at time t_i, called the time stamp of the event. In packet communication networks, the relevant events are the sending, arrival and generation of packets.

Although BLOSIM provides only a time-driven simulation environment, it can be extended to handle event-driven simulations, with the addition of an event sequence manager implemented as a BLOSIM block. The event sequence manager ensures that the sequence of events in the network are simulated according to the ordering of their respective time stamps, while allowing loosely coupled local clocks at the network nodes. Early in this effort, this approach was implemented to simulate loop topology store-and-forward packet communication networks. We found that the additional execution overhead required by the event sequence manager created excessive simulation execution times. We subsequently developed a stand-alone dedicated simulation software shell in Pascal for general store-and-forward packet communications networks, which provided substantially faster execution times. This software shell was subsequently used to develop dedicated software modules for simulating the various configurations of the store-and-forward networks and the multiple-access networks provided in the network model library. The simulation operating system is then reduced to loading the executable

simulation code for the user specified configuration. This approach was feasible because of the small number of executable code modules required due to the homogeneous nature of communication networks. It is not feasible for simulating communication links because the large number of executable code modules there requires an exorbitant amount of hard disk storage.

The network simulation architecture is as follows. The network portion of the model library consists of the executable simulation code modules for each network configuration. The interface software loads the executable code for the user specified configuration to run the simulation. Animated graphics for viewing during the simulation run is switched on or off in the executable code according to user specifications. Post-run displays are accessed through the interface software, which loads the appropriate executable graphics generating routines.

3. INTERFACE SOFTWARE

The interface software performs the following functions.

1) Provides mouse-driven "Lotus Symphony-like" slash-bar type menu user interface for system configurations.

2) Determines the validity of the user specified configuration.

3) Generates system configuration parameter data files for passing the parameters to the executable simulation code.

4) In the link simulation environment, it generates the UNIVERSE C source code from the user configuration and initiates the necessary compilation and linking to obtain the executable simulation code. In the network environment, it loads the executable simulation code for each individual configuration.

5) Initiates the simulation run from user command.

6) Provides the mouse-driven slash-bar menu interface to the post-processor display functions.

The interface software was written in 8086 assembly language in order to minimize execution time. The slash-bar menu interface has a linked list data structure in which each list contains the following information:

a) The address of the current layer command list.
b) The adresses of the next layer lists.
c) The addresses of the next layer function calls.
d) List Control parameters.

The interface contains a numeric editor that controls the keyboard entered numerical parameters. An artificially intelligent rule and table based interpreter checks the validity of the user specified configuration. Finally, it contains a module which generates the UNIVERSE C source code from the user specified link configuration. The interface software source code is provided in the two files LN.ASM and LN1.ASM. The source code listing is given in Volume III of this report.

The remainder of this section is a description of the user operation of the interface and serves as a user manual. The interface software controls the entire visual simulation system. The software is loaded and execution begun by entering from the keyboard the command

Page 7 not included

3.2 Link

Executing LINK in the main menu brings up the menu shown in Figure 3. The upper slash-bar indicates the history MAIN MENU followed by LINK. Executing MAIN MENU returns there. The lower slash-bar contains the choices:

CONFIGURATION - To enter system configuration.

SIMULATE - To select the maximum simulation time and/or to run the simulation.

POST_PROCESSING_FILES - To view post-run simulation results.

QUIT - To return to the main menu.

3.2.1 Configuration

Executing CONFIGURATION brings up the menu and communications link system block diagram screen shown in Figure 4. System configurations can be saved and retrieved. The upper slash-bar indicates the history MAIN MENU followed by LINK followed by CONFIGURATION. Executing LINK returns to the last menu and executing MAIN MENU returns to the first menu. The choices in the lower slash-bar are:

RETRIEVE - To retrieve a system configuration. As shown in Figure 5a, the user is prompted for the specification of the file containing the desired system configuration data. An invalid or non-existent file specification results in the screen shown in Figure 5b. Here the mouse input is activated by clicking either button. An attempt to retrieve a file which is not a system configuration data file will result in the screen shown in Figure 5c. These user friendly help messages ensure that the interface software will not crash under inadvertent misuse.

SAVE - To save the current system configuration in a disk file. Any valid MSDOS filename is allowable for a system configuration file. The screens are identical to that for RETRIEVE.

EDIT - To edit the current system configuration. This choice is also selected whenever a new configuration is initially set up.

DOS - Same function as in MAIN_MENU.

QUIT - Return to MAIN_MENU.

Execution of EDIT results in the screen shown in Figure 6, which displays the communication link system block diagram and choices

for configuring each of the blocks. When a block is configured, the actual functional block choice (for example BPSK in the MODULATOR block) is displayed in the system block diagram. The choices are:

SOURCE - Execution results in the screen shown in Figure 7, which displays the two possible choices for the Source block: RANDOM_BIT_STREAM and SINUSOID. The user is prompted to enter the data rate for the Random Bit Stream choice; and the sinusoidal and sampling frequencies for the Sinusoid choices. The sinusoid choice is intended as a source input for simulating the filters by themselves.

ENCODER - Execution results in the screen shown in Figure 8, which displays the three possible choices for the Channel Encoder Block: CONVOLUTIONAL, REED_SOLOMON or NONE. Execution of the CONVOLUTIONAL choice prompts the user to choose between the binary rate 1/2 constraint length 3 and 7 codes. Execution of the REED_SOLOMON choice results in the (15,9) code. The NONE choice is used for uncoded systems.

MODULATOR - Execution results in the screen shown in Figure 9, which displays the four possible choices for the Modulator block: PSK, QAM, FSK and NONE. The PSK choice prompts the user to choose among the signal vector BPSK, QPSK, 8PSK modulation or the time-domain BPSK modulation cases. In the other signal vector FSK or QAM modulation cases, the user is prompted to choose among the symbol alphabet sizes (2 or 4 for FSK and 16 or 64 for QAM). The NONE choice can be used when only the filters by themselves are simulated.

TRANSMITTER_FILTER - Execution results in the screen shown in Figure 10, which displays the five choices for the Channel Filter Block: BUTTERWORTH, CHEBYCHEV, ELLIPTIC, FIR and NONE. The Butterworth, Chebychev IIR filters allow specification in terms of sampling frequency, 3db cutoff frequency and order only; or in terms of passband and stopband attenuations and edge frequencies and sampling frequency. The Elliptic IIR filter allows specification only in terms of the passband and stopband attenuations and edge frequencies and sampling frequency. The FIR filter allows specification in terms of the order and the filter coefficients; or in terms of passband and stopband attenuations and edge frequencies and sampling frequency. For example, the display screen which allows the user to enter and edit FIR coefficient values is shown in Figure 11 for an order 99 filter. An additional feature is the SAVE_FIR choice, which saves the user entered FIR filter coefficients in a disk file. Any valid MSDOS filename is allowable for this file. These coefficients can subsequently be retrieved from that file using the RETRIEVE_FIR choice.

CHANNEL - Execution results in the screen shown in Figure 12, which displays the three choices for the Channel Block: WHITE GAUSSIAN NOISE, PROPAGATION MODEL and BSC. Again, the user is prompted for the appropriate parameters in each choice.

RECEIVER_FILTER - Same as TRANSMITTER_FILTER.

DEMODULATOR - The available choices are SOFT or HARD decisions demodulation or NONE as shown in Figure 13.

DECODER - The available choices are CONVOLUTIONAL (with a subsequent prompt to select either HARD_DECISION_VITERBI or SOFT_DECISION_VITERBI decoding), REED SOLOMON decoding or NONE. THis is shown in Figure 14.

QUIT - Return to the MAIN_MENU.

3.2.2 Simulate

Executing SIMULATE brings up the screen shown in Figure 15a. The lower slash-bar in this menu has the choices RUN and QUIT. The choice QUIT again returns to the MAIN_MENU. The choice RUN prompts the user to enter the maximum simulation time desired in terms of the number of samples. Pressing the ENTER key then directs the interface software to execute the simulation run. For a new configuration, the interface software will first check whether it is a valid and completely specified configuration. An incomplete configuration will result in the screen shown in Figure 15b. Invalid configurations bring up a similar screen. The interface software will proceed to prepare and execute the simulation of a valid configuration. This consists of the following steps, the progress of which is displayed on the screen.

1) Parameter data files are first generated.
2) The UNIVERSE C source code (file UNI.C) is generated from the configuration.
3) The Definicon loader LOAD.COM is executed to load the SVS C compiler shell CC.E20 to compile UNI.C.
4) LOAD.COM is executed again to load the SVS linker LINK20.E20 to link the UNIVERSE object code with the BLOSIM run-time object code and the required functional block object code modules to generate the executable code UNI.E20.
5) LOAD.COM is executed to load UNI.E20 to begin the simulation run.

In the event of a configuration which has just been simulated, only the first step 1) and the last step 5) are performed. So the compiling and linking processes are not required when only parameter values in a configuration are changed for the subsequent simulation run. The simulation run can be suspended by pressing the PAUSE key. Pressing the space bar will resume the

run. Also the run can be permanently aborted by pressing the CTRL-BREAK key.

3.2.3 Post Processing Files

Executing POST_PROCESSING_FILES brings up the screen shown in Figure 16. The lower slash-bar menu has the choices VIEW_DATA and VIEW_GRAPH. Executing VIEW_DATA brings up the screen shown in Figure 17a with lower slash-bar menu choices of PERFORMANCE_DATA, TRANSMITTER_FILTER_DESIGN_DATA, RECEIVER_FILTER_DESIGN_DATA, and QUIT. The QUIT choice operates similar to previous menus. The PERFORMANCE_DATA choice can be executed only for configurations with the random bit stream source. This choice displays on screen the final simulation bit and symbol error probability results. The filter design data choices can be executed only for configurations involving the respective filters. These choices display the designed filter coefficients for IIR filters and the filter impulse response for FIR filters. Appendix B gives the names of the disk files containing the displayed data in these three choices. These data files can be transferred to a printer for hard copy under the MSDOS operating system. Executing VIEW_GRAPH brings up the screen shown in Figure 17b with lower slash-bar menu choices of BIT_ERROR_RATE, SIGNAL_CONSTELLATIONS, FILTER_WAVEFORMS and QUIT. These choices allow the user to access the corresponding post-run graphics displays. The choice BIT_ERROR_RATE can be executed only for configurations with the random bit stream source. Executing this choice brings up the screen shown in Figure 18. The mouse can be used to move the cursor to the highlighted portions of the screen to either view the probability of error versus simulation time plot or to return to the previous menu. The function keys F1 and F2 serve similar purposes. The choice SIGNAL_CONSTELLATIONS can be executed only for modulations with signal space dimension greater than one. Executing this choice brings up the screen shown in Figure 19. Again either the mouse or the function keys can be used to view the signal constellation plots. The choice FILTER_WAVEFORMS can be executed only for configurations involving filters. Executing this choice brings up the screen shown in Figure 20. Again either the mouse or the function keys can be used to view the respective filter input or output waveforms. Note that the system block diagram in Figures 18, 19, and 20 display the current simulation configuration.

3.3 Network

This section explains operations in the communications network environment. Since the execution of the NETWORK menus is similar to that of the LINK menus in many respects, only the differences between the two environments will be explained.

Executing NETWORK in the main menu brings up the menu shown in Figure 21. The upper slash-bar indicates the history MAIN

MENU followed by NETWORK. The lower slash-bar contains the choices:

CONFIGURATION - To enter system configuration.

SIMULATE - To select the simulation run parameters and/or to run the simulation.

POST-PROCESSING FILES - To view post-run simulation results.

QUIT - To return to the main menu.

3.3.1 Configuration

Executing CONFIGURATION brings up the display screen shown in Figure 22. System configurations can be saved and retrieved. The upper slash-bar indicates the history MAIN MENU followed by NETWORK followed by CONFIGURATION. Executing NETWORK returns to the last menu and executing MAIN MENU returns to the first menu. Execution of RETRIEVE, SAVE, DOS, and QUIT choices are identical to that in the communications link environment. Execution of EDIT results in the screen shown in Figure 23 which displays the choices of either configuring the store-and-forward network or the multiple access radio network. The last choice QUIT in the lower slash-bar menu again returns to the MAIN_MENU.

3.3.1.1 Store and Forward Network

Executing STORE_AND_FORWARD results in the screen shown in Figure 24, which displays the four configuration parameters that must be specified for the store and forward network. The current version of the simulation software allows only the shortest path routing algorithm. The four configuration parameters are:

TOPOLOGY - Executing TOPOLOGY results in the screen shown in Figure 25, which displays the two choices: STAR and LOOP. The STAR choice selects a star network topology while the LOOP choice prompts the user for either the uni-directional loop or the bi-directional loop as shown in Figure 26a. Selecting STAR results in Figure 26b, and subsequently selecting LINK_CAPACITIES results in Figure 26c, which prompts user for the capacity of the links in the star topology. In the STAR topology, node 1 always represents the central node. Similarly, the UNI_DIRECTIONAL_LOOP choice results in the screens shown in Figures 26d and 26e while the BI_DIRECTIONAL_LOOP choice will display screens shown in Figures 26f and 26g. The number of links that the user is prompted depends on the NUMBER_OF_NODES and the TOPOLOGY specified. After specifying the link capacities for either topology, the user will be returned to the STORE_AND_FORWARD menu for specifying other configuration parameters. The current selection for each parameter is displayed on the screen. Link capacities are specified in terms of packets per second.

NUMBER__OF_NODES - Executing NUMBER_OF_NODES results in the screen shown in Figure 27, which prompts for the number of nodes. This should be the first parameter entered by the user. The maximum allowable number of nodes is 8 for store-and-forward networks.

INPUT_TRAFFIC_MATRIX - Executing INPUT_TRAFFIC results in the screen shown in Figure 28 (this screen is an 8-node network example), and prompts the user to enter the input traffic matrix. Input traffic is specified in terms of packets per second. The row index is the origination node number and the column index is the destination node number.

REAL_TIME_ANIMATION - Executing REAL_TIME_ANIMATION results in the screen shown in Figure 29, and prompts the user to select either ON or OFF. Selecting either one will return the user to the previous screen with the type of selection shown on the screen.

QUIT - return to the MAIN_MENU.

3.3.1.2 Multiple Access Radio Network

Executing MULTIPLE_ACCESS results in the screen shown in Figure 30, which displays the five configuration parameters that must be specified for the multiple access radio network. The choices are:

PROTOCOL - Executing PROTOCOL results in the screen shown in Figure 31a, which displays the three configurations that can be specified: ALOHA, TREE, and CSMA. Selecting ALOHA will result in the screen shown in Figure 31b which prompts the user for the MAXIMUM_BACKOFF_DELAY in terms of number of slots. Selecting CSMA will result in the screen shown in Figures 31c, which prompts the user for the MAXIMUM_BACKOF-F_DELAY and PROPAGATION_RATIO. The MAXIMUM_BACKOFF_DELAY is specified in terms of slots. The PROPAGATION_RATIO is specified by entering the number of mini-slots per slot where each mini-slot is equal to the round-trip propagation delay. The usual CSMA propagation ratio is then the inverse of this number. After specifying these parameters, the user will be returned to the previous menu. Again, the current selection for each parameter is displayed on the screen. Selecting TREE will not prompt the user for any input.

USER_POPULATION - Executing USER_POPULATION prompts for the number of users in the network as shown in Figure 32. The current maximum is 20 stations. Each station generates Poisson new packet arrivals.

NUMBER_OF_CHANNELS - Executing NUMBER_OF_CHANNELS results in the display shown in Figure 33, which prompts for the number of channels in the network. The current maximum is 1 for CSMA and TREE, and 10 for ALOHA.

INPUT_TRAFFIC_MATRIX - Executing INPUT_TRAFFIC_MATRIX results in a screen similar to that shown in Figure 28, which prompts the user to input the traffic matrix. The input traffic rate is specified in terms of packets per slot per channel for each station.

REAL_TIME_ANIMATION - Executing REAL_TIME_ANIMATION prompts the user for turning the real time animation ON or OFF similar to the store-and-forward network case as shown in Figure 29.

QUIT - return to the MAIN_MENU.

3.3.2 Simulate

Executing SIMULATE brings up the screen shown in Figure 34. The lower slash-bar in this menu has the choices RUN, OUTPUT_DISPLAY and QUIT. The choice RUN prompts the user to enter the simulation time. The time units are in terms of seconds for store-and-forward networks and in terms of slots for multiple access networks. The simulation is then executed for that length of time. The choice QUIT again returns to the MAIN_MENU. Execution of the choice OUTPUT_DISPLAY results in the screen shown in Figure 35 with choices STORE_AND_FORWARD and MULTIPLE_ACCESS. This menu allows the user to specify these displays to be active during the simulation run when REAL_TIME_ANIMATION is turned ON. Executing the choice STORE_AND_FORWARD gets the screen shown in Figure 36. The mouse can be used to toggle the choices ON or OFF. Only one of the choices can be specified either ON or OFF. The user is prompted to specify the origination and destination nodes when turning either choice ON. If these nodes are not specified, a default will be selected by the software during the simulation run. Executing FINISH will return the user to the OUTPUT_DISPLAY menu. The F2 and F3 function keys can also be used to select the display desired and the F1 key to finish the selection. Similarly, Figure 37 shows the screen following execution of the MULTIPLE_ACCESS choice. When the REAL_TIME_ANIMATION is turned OFF during a simulation run, the run can be suspended by pressing the PAUSE key and aborted by pressing the CTRL-BREAK key, similar to the Link environment. However, the run cannot be suspended or aborted when REAL_TIME_ANIMATION is turned ON.

3.3.3 Post Processing Files

Executing POST_PROCESSING_FILES brings up the screen shown in Figure 38. The lower slash-bar menu has the choices VIEW_DATA and VIEW_GRAPH. Executing VIEW_DATA displays the average packet delay and average throughput simulation results. Appendix B gives the filename of the disk file containing the displayed data. Executing VIEW_GRAPH brings up the screen shown in Figure 39 with lower slash-bar menu choices of PACKET_DELAY and THROUGHPUT. These choices allow the user to access the corresponding post-run graphics displays.

4. LINK MODEL LIBRARY

This section describes the capability of the link model library and explains the user specified parameters in the link configuration process. We first describe the set of valid configurations supported by this library and indicate some sample configurations files supplied on the LINKNET Visual Simulation Software distribution disks (see Appendix A). The total number of 175 valid configurations includes 100 configurations involving modulations, 3 binary symmetric channel configurations, and 72 filters only configurations. Sample configuration files are not supplied for every possible configuration. The capability and characteristics of each functional block are then described in detail.

4.1 Supported Link Simulation Configurations

The first set of supported configurations involve uncoded modulation without filtering over the additive white Gaussian noise channel. The .LIK files are the corresponding available sample configurations supplied with the software (see Appendix A).

1) Uncoded Coherent BPSK - UNCBP.LIK
2) Uncoded Coherent QPSK - UNCQP.LIK
3) Uncoded Coherent 8PSK - UNC8P.LIK
4) Uncoded Coherent 16QAM - UNC16QAM.LIK
5) Uncoded Coherent 64QAM - UNC64QAM.LIK
6) Uncoded Noncoherent BFSK - UNCBF.LIK
7) Uncoded Noncoherent 4FSK - UNCQF.LIK

The second set involve convolutional coded modulation over the additive white Gaussian noise channel with hard decision Viterbi Decoding.

8) (1/2,3) Coherent BPSK - BPHRD3.LIK
9) (1/2,7) Coherent BPSK - BPHRD7.LIK
10) (1/2,3) Coherent QPSK - QPHRD3.LIK
11) (1/2,7) Coherent QPSK - QPHRD7.LIK
12) (1/2,3) Noncoherent BFSK - BFHRD3.LIK
13) (1/2,7) Noncoherent BFSK - BFHRD7.LIK

The next set involve convolutional coded modulation over the additive white Gaussian noise channel with soft decision Viterbi Decoding.

14) (1/2,3) Coherent BPSK - BPSFT3.LIK
15) (1/2,7) Coherent BPSK - BPSFT7.LIK
16) (1/2,3) Coherent QPSK - QPSFT3.LIK
17) (1/2,7) Coherent QPSK - QPSFT7.LIK

The remaining configurations involving transmission over the additive white Gaussian noise channel are:

18) (15,9) Reed Solomon coded 16QAM - 16QAMRS.LIK

19) Uncoded Filtered Coherent BPSK - BPFIR.LIK (FIR filters)
20) Uncoded Filtered Time-Domain BPSK - TBPFIR.LIK (FIR filters), TBPBUT.LIK (Butterworth filters), TBPCHEB.LIK (Chebychev filters), TBPELL.LIK (Elliptic filters)

All the above configurations 1) -20) are also valid with the propagation model channel in place of the additive white Gaussian noise channel. The sample configuration PROP.LIK is uncoded coherent BPSK over the propagation model channel. The following set of configurations involve the Binary Symmetric Channel (BSC).

21) Uncoded BSC - UNCBSC.LIK
22) (1/2,3) Convolutional Coded BSC - BSC3.LIK
23) (1/2,7) Convolutional Coded BSC - BSC7.LIK

There are six valid generic filter configurations with no modulation and no coding. All of these configurations allow any choice of filter and use the sinusoidal source. Three of the six generic configurations allow either the additive white Gaussian noise channel or the propagation model channel. The other three involve no channel. The six generic configurations and the supplied sample configurations (not every possible sample configuration is supplied) are:

24) Transmitter and Receiver Filter with Channel - FIR1.LIK (FIR filter)
25) Transmitter Filter with Channel - FIR2.LIK (FIR filter)
26) Channel with Receiver Filter - FIR3.LIK (FIR filter)
274) Transmitter and Receiver Filter, no Channel - FIR4.LIK (FIR filter)
25) Transmitter Filter, no Channel - FIR5.LIK (FIR filter)
26) Receiver Filter, no channel - FIR6.LIK (FIR filter), BUT6.LIK (Butterworth filters), CHEB6.LIK (Chebychev filters), ELL6.LIK (Elliptic filters)

4.2 Signal generator Modules

The following blocks are signal sources.

	Block	C Function	File
1)	Random Bit Stream	gen()	GEN.C
2)	Sinusoidal Source	gen()	SIN.C

The C function block gen() given in file GEN.C generates equiprobable random bit streams. The algorithm used is based on the linear congruential method. The user is prompted for the bit rate in terms of bits per second (bps). The C function block gen() in file SIN.C generates a unit amplitude discrete-time sinusoidal signal with user specified sinusoidal frequency and sampling frequency in Hz. In order to obtain a constant amplitude discrete sinusoidal signal source, the user is cautioned to select a sampling frequency that is divisible by the sinusoidal

frequency. Otherwise amplitude modulation effects will result from the asynchronous sampling. The sampling frequency should also be identical to the filter sampling frequencies, and must be at least twice the sinusoidal frequency.

4.3 Digital Modulator and Demodulator Modules

The following blocks implement the modulators and hard decision demodulators for some standard digital modulations.

Block	C Function	File
1. Coherent BPSK Modulator	bpsk_mod()	BPMOD.C
2. Coherent BPSK Demodulator	bpsk_dem()	BPDEM.C
3. Coherent QPSK Modulator	qpsk_mod()	QPMOD.C
4. Coherent QPSK Demodulator	qpsk_dem()	QPDEM.C
5. Coherent 8-PSK Modulator	psk8_mod()	8PMOD.C
6. Coherent 8-PSK Demodulator	psk8_dem()	8PDEM.C
7. Coherent 16-QAM Modulator	qam16_mod()	16QMOD.C
8. Coherent 16-QAM Demodulator	qam16_dem()	16QDEM.C
9. Coherent 64-QAM Modulator	qam64_mod()	64QMOD.C
10. Coherent 64-QAM Demodulator	qam64_dem()	64QDEM.C
11. Non-coherent BFSK Modulator	bfsk_mod()	BFMOD.C
12. Non-coherent BFSK Demodulator	bfsk_dem()	BFDEM.C
13. Non-coherent 4-FSK Modulator	qfsk_mod()	QFMOD.C
14. Non-coherent 4-FSK Demodulator	qfsk_dem()	QFDEM.C
15. Soft Decisions Demodulator	no_dem()	NO_DEM.C

In these modules, modulated signals are represented in the equivalent signal vector domain. Coherent matched filter demodulators and non-coherent envelope detector demodulators are implemented. The non-coherent signalling environment assumes uniformly distributed random phases. The constant energy signal modulations all have unit energy. The 16QAM signal co-ordinates along each axis are -3, -1, +1, +3; and the 64QAM signal co-ordinates are -7, -5, -3, -1, +1, +3, +5, +7. Signal constellations data files are generated in the demodulators for modulations with signal space dimension greater than one. These data files are used by the post-run signal constellation graphics generating routine.

Signal vector simulations were chosen to minimize simulation execution time. The library also contains the implementation of Coherent BPSK modulation and demodulation in the time domain. Here the equivalent baseband time domain BPSK modulation is implemented with a 20X oversampling. The modulator outputs rectangular pulses of amplitude ± 1. The demodulator bases its decision on samples at the middle of the symbol period. The blocks are:

Block	C Function	File
16. Time Domain BPSK Modulator	bpsk_mod()	TBPMOD.C
17. Time Domain BPSK Demodulator	bpsk_dem()	TBPDEM.C

The time-domain BPSK modulation is intended for use with transmitter and receiver filters. The demodulator uses the filter order for IIR filters and half the filter order for FIR filters to estimate filter delays. These estimates are exact only in the case of linear phase FIR filters. So in configurations other than those involving linear phase FIR filters, the resulting error probability could reflect synchronization error when the filter order exceeds 20.

4.4 Transmitter and Receiver Filter Modules

The following blocks implement transmitter and receiver filters.

Block	C Function	File
1. Transmitter Butterworth	txlpf()	TXBUT.C
2. Transmitter Chebychev	txlpf()	TXCHEB.C
3. Transmitter Elliptic	txlpf()	TXELL.C
4. Transmitter FIR	txlpf()	TXFIR.C
5. Receiver Butterworth	rclpf()	RCBUT.C
6. Receiver Chebychev	rclpf()	RCCHEB.C
7. Receiver Elliptic	rclpf()	RCELL.C
8. Receiver FIR	rclpf()	RCFIR.C

The IIR Butterworth, Chebychev and Elliptic lowpass digital filters are obtained by a bilinear transformation of the corresponding analog transfer function. The user can either specify: sampling frequency, passband edge frequency, stopband edge frequency, passband attenuation and stopband attenuation; or specify the sampling frequency, order and 3dB cutoff frequency only for the Butterworth and Chebychev cases. The blocks perform

the required digital filter design. Only the first user option of specifying the sampling frequency, passband and stopband edge frequencies, and the passband and stopband attenuations is available for the Elliptic filter. In all three blocks, cascade filter realization of second order sections was used. For the FIR lowpass digital filters, the user can either specify: sampling frequency, passband edge frequency, stopband edge frequency, passband attenuation and stopband attenuation; or specify the sampling frequency, order, cutoff frequency and filter coefficients directly. Filter synthesis in the first option was accomplished by using the Kaiser windowing method. The designed filter has a linear phase characteristic. The user is reminded that the sampling frequency of the filters must be at least twice the passband and stopband edge frequencies and the cutoff frequency.

The designed filter coefficients in the IIR case and the designed filter impulse response in the FIR filter case can be viewed from the VIEW_DATA option in the POST_PROCESSING_FILES menu. For IIR filters, the coefficients of each of the cascade sections are given. The i-th second order section has transfer function

$$\frac{a_0[i] + a_1[i]\ z^{-1} + a_2[i]\ z^{-2}}{1 + b_1[i]\ z^{-1} + b_2[i]\ z^{-2}}$$

and the first order section (if any) has transfer function

$$\frac{a_0[0] + a_1[0]\ z^{-1}}{1 + b_1[0]\ z^{-1}}$$

The overall filter tranfers function is then the product of the tranfer functions of all the sections times a gain constant H_o. For FIR filters, the impulse response {h[n]} is given. Moreover the filter input and output waveforms can be viewed after a simulation run from the VIEW_GRAPH option.

4.5 Channel Encoder and Decoder Modules

The following blocks implement error-correction encoders and decoders.

Block	C Function	File
1. (1/2,3) Convolutional Encoder	conv()	CONV3.C

2. (1/2,7) Convolutional Encoder	conv()	CONV7.C
3. (15,9) Reed Solomon Encoder	rscod()	RSCOD.C
4. No encoder	nocod()	NOCOD.C
5. Hard Decisions Viterbi Decoder (1/2,3) code	viterbi()	VIT3.C
6. Hard Decisions Viterbi Decoder (1/2,7) code	viterbi()	VIT7.C
7. Soft Decisions Viterbi Decoder (1/2,3) code	viterbi()	VIT3Q.C
8. Soft Decisions Viterbi Decoder (1/2,7) code	viterbi()	VIT7Q.C
9. Reed Solomon Decoder	rsdec()	RSDEC.C
10. No decoder	nodec()	NODEC.C

The rate 1/2 constraint length K=3 binary convolutional code has code generators 5,7 in octal. This is the best constraint length 3 rate 1/2 code known and has a free distance of 5. The corresponding hard decisions Viterbi decoder also serves as the soft decisions decoder for coherent BPSK modulation only. The rate 1/2 constraint length K=7 binary convolutional code has code generators 171,133 in octal and is the optimal Odenwalder code for this rate and constraint length. This code has a free distance of 10. The corresponding hard decisions Viterbi decoder also serves as the soft decisions decoder for coherent BPSK modulation only. The soft decisions decoders are for coherent QPSK modulation only. The Viterbi decoders have a path memory of 32 bits. This allows 5 constraint lengths of memory for the constraint length K=7 code and even more for the constraint length K=3 code.

The encoder for the (15,9) Reed Solomon code is a systematic encoder in $GF(2^4)$ with generator polynomial

$$g(x) = \alpha^6 + \alpha^9 x + \alpha^6 x^2 + \alpha^4 x^3 + \alpha^{14} x^4 + \alpha^{10} x^5 + \alpha^0 x^6,$$

where α is the primitive element of $GF(2^4)$ that is the root of the primitive polynomial

$$x^4 + x + 1$$

over GF(2). This code has $d_{min} = 7$ and corrects any pattern of up to 3 errors. The corresponding Reed Solomon decoder first computes the syndrome and then obtains the error-locator polynomial by using the modified Euclid algorithm. It then computes the error sequence in the $GF(2^4)$ Discrete Fourier Transform domain

and finally performs an inverse transform to obtain the decoded error pattern.

The nocod() block is used to relay the random bit stream to the display() block (see Sink Modules below) for the computation of error probabilities. The nodec() block is included for consistency in generating the UNIVERSE.

4.6 Channel Modules

The following blocks are used to implement channels.

Block	C Function	File
1. Adder	adder()	ADD.C
2. White Gaussian Noise	gauss()	GAUSS.C
3. Propagation Model	gauss()	PROP.C
4. Uncoded BSC	bsc()	BSC.C
5. Coded BSC	bsc()	CBSC.C

The block gauss() given in file GAUSS.C generates a zero mean white Gaussian noise sequence and is based on a transformation of the linear congruential method. Additive white Gaussian noise channels are implemented using this block along with the adder() block that adds the white noise sequence to the transmitted signal samples. In those configurations involving additive white Gaussian noise channels, the user enters the received signal power to noise power density ratio S/N_o. For the signal vector implemented modulation configurations, the gauss() block then calculates the resulting signal energy to noise power density ratio E_s/N_o and the bit energy to noise power density ratio E_b/N_o and adjusts the noise sample variances accordingly. These performance measures are displayed on screen during the simulation run and also may be viewed from the PERFORMANCE_DATA option. For the time-domain BPSK configuration, the user inputed S/N_o is assumed to be at the receiver filter input and the noise bandwidth is assumed to be equal to the transmitter cutoff frequency. For the configurations involving filters only, the noise bandwidth is assumed to be equal to the sampling frequency.

The block gauss() given in file PROP.C also generates a zero mean white Gaussian noise sequence and is used along with the adder() block. It is intended however to implement an additive white Gaussian noise channel in a ground mobile radio propagation loss environment. The propagation loss is estimated using a standard area-to-area path loss prediction model. In this model, the propagation loss is determined from the following user specified parameters:

1. Transmitter antenna height (in meters)
2. Receiver antenna height (in meters)
3. Transmitter antenna gain (in db)
4. Receiver antenna gain (in db)
5. Carrier frequency (in MHz)
6. Receiver bandwidth (in Hz)
7. Distance between transmitter and receiver (in Km.)
8. Receiver noise power (in dbm)
9. Area type: open or suburban
10. Foliage characteristic: foliaged or no foliage

The formula used to determine path loss is given by Lee [2]. The standard deviation of the predicted path loss is 8db. This propagation loss formula then determines the received signal power to noise power density ratio S/N_o. The rest of this block then performs the same function as the additive white Gaussian noise block described above, using this calculated value of S/N_o rather than the user specified value as in the white Gaussian noise block.

Also included are the blocks bsc() given in files BSC.C and CBSC.C, which implements a binary symmetric channel for the uncoded and convolutional coded cases respectively. The user enters the raw channel bit error probability in these blocks.

4.7 Sink Modules

The following blocks are sinks in the communication system.

Block	C Function	File
1) Display Error Probability	display()	DUNC.C
2) Display Error Probability for convolutional coded systems	display()	DCONV.C
3) Sink	sink()	SINK.C

The statistic of primary interest in evaluating data communications link performance are the bit and symbol error rates. The block display() given in file DUNC.C performs the calculation of sample-averaged bit and symbol error rates between data source and sink for uncoded and Reed Solomon coded systems. The block display() in file DCONV.C performs the same function for convolutional coded systems. These blocks provide an animated display of the running bit and symbol error probability during the simulation run. Final error probabilities are accessible from the PERFORMANCE_DATA option. A graphical display of the dynamical bit error probability behavior is also available in the VIEW_GRAPH option. The sink block is required for consistency in running BLOSIM for configurations involving only the filters.

5. NETWORK MODEL LIBRARY

This section describes the capability of the network model library and explains the user specified parameters in the network configuration process. We describe the principles of operation of the model library. Some sample network configuration files are supplied on the LINKNET Visual Simulation Software distribution disks (see Appendix A). The sample network files have identical file names as the source code file name except that the file extension .PAS is replaced by .NET. For example, the sample configuration of a slotted ALOHA network is given in the file ALOHA.NET. The user can turn the real time animation capability ON or OFF before a simulation run. However it is cautioned that turning on the animation increases the execution time tremendously. This is due to the slow IBM PC/XT graphics display hardware. A worthwhile Phase II development effort is to employ a dedicated high-speed graphics processor board for animated screen displays.

5.1 Store-and Forward Network Modules

The store-and-forward network modules, STARNET.PAS, UNILOOP.PAS, and BIDLOOP.PAS, simulate the star, uni-directional loop and bi-directional loop topologies respectively. We have constructed a general store-and-forward network simulation software shell which provides the skeleton for all three modules. This software shell can simulate any size and topology store-and-forward type packet-switched network by specifying three matrices, the Link Matrix, Session Matrix and Routing matrix. The above modules are special cases of this general software shell for the respective topologies. Each node in the network is assigned a unique number. The format of each matrix is described as follows.

A. Link Matrix: The Link matrix specifies the beginning node number, the end node number, and the capacity of each link in the store-and-forward network. Each link is assigned a channel_number used by the other matrices for cross referencing. The format of the Link matrix is given by

	from	to	capacity
channel_number	node_number	node_number	bits/sec

B. Session Matrix: The Session matrix specifies the origination node number, the destination node number, the traffic requirement and the mean packet length of each traffic session in the store-and-forward network. Each session is assigned a session_number used by the

Routing matrix for cross referencing. The format of the Session matrix is given by

	origin	destination	traffic_rqmt	mean_pkt_length
session_number	node_number	node_number	packets/sec	bits/packet

C. Routing matrix: The routing matrix specifies the out-bound channel_number for each node and session in the network. The node processing time is also in included in this matrix. The format of the Routing matrix is given by

	session_number		node_process_time
node_number	out_bound_queue	channel_number	seconds

The general store-and-forward simulation software shell poses no limitation on the number of network nodes, the number of links in the network, and the number of parallel sessions in the network. We artificially impose some limits on these parameters so that the parameters can be fitted on the size of a PC type screen. For all three topologies specified by the user, the corresponding software will build these three matrices based on the user supplied input configurations. Following is a description of each topology. The curent release of the simulation software does not allow the user to specify the node_process_time. It is set to zero.

5.1.1 Star Topology

In the star topology, the hub node number is assigned node number 1. The rest of the nodes are assigned by the software automatically.

5.1.2 Uni-Directional Loop Topology

In a uni-directional loop topology, all node numbers are assigned by the software automatically. The direction of each link is shown in the parameter input screen.

5.1.3 Bi-Directional Loop Topology

In a bi-directional loop topology, all node numbers are assigned automatically. The software computes the shortest path between any pair of nodes to build the Routing matrix.

5.2 Multiple Access Network Modules

The multiple access network modules, ALOHA.PAS, TREE.PAS, and CSMA.PAS, representing the slotted ALOHA, non-persistent slotted CSMA, and the blocked-access Tree Collision Resolution Algorithm protocols respectively. All three modules assume Poisson packet arrival process. The total user population arrival rate is the sum of the arrival rates of all the stations. Input parameters are different for each module.

5.2.1 ALOHA Module

This module implements the multiple channel slotted-ALOHA protocol. A slot time is equal to the packet transmission time. The maximum number of channels is limited to 10 in the current release, although it can be increased by changing one parameter in the declaration section of the source code ALOHA.PAS. The input parameter MAXIMUM_BACKOFF_DELAY specifies the maximum allowable backoff delay for stabilizing the protocol in terms of the number of slots. A value of 15 slots is recommended for most applications. The capacity is 0.368 packets per slot per channel. The user is cautioned against performing simulations at traffic rates approaching capacity since this results in unstable behavior.

5.2.2 TREE Module

This module implements the Capatenakas Tree Algorithm with the blocked channel access procedure. Only one channel is allowed in the current release. LINKNET is currently performing work which generalizes this protocol to multiple channels. No input parameters are required for this protocol other than the input rate traffic specification. This protocol has a stable capacity of 0.347 packets per slot.

5.2.3 CSMA Module

This module implements the non-persistent slotted CSMA protocol. Only one channel is allowed in the current release. In this protocol, a slot time is equal to the packet transmission time while a mini-slot time is equal to the round trip propagation delay. There are two parameters that can be specified for this protocol. The MAXIMUM_BACKOFF_DELAY specifies the maximum allowable backoff delay for stabilizing the protocol in terms of the number of slots. The PROPAGATION_RATIO specifies the ratio of the propagation delay to the packet transmission time in terms of the number of mini-slots per slot. For most applications, a MAXIMUM_BACKOFF_DELAY of 15 slots and PROPAGATION_RATIO of 100 mini-slots are reasonable choices.

6. SIMULATION OUTPUT GRAPHICS

The graphics software was developed using SCI-GRAF modules, a library of plotting and graphics procedures available for use with the DSI-750 Silicon Valley Software "C" and PASCAL compilers. This library supports the Hercules graphics card acquired with the Definicon DSI-750 co-processor board enhanced Compaq Deskpro computer. The SCI-GRAF library of high level procedures allows the creation of plots of curves while its low level procedures allows point plots at pixel coordinates.

Display of signal constellation plots is available from the VIEW_GRAPH option for link configurations with signal set dimensions greater than one. Samples of the soft decisions channel demodulator outputs are displayed in a scatter plot depicting the clusters around the signal vectors generated by channel noise. Figure 40 shows such a plot for a simulation of an uncoded QPSK system. Each signal constellation plot requires 2000 sample points. Therefore the maximum simulation time must be at least 2000 to obtain a plot. The I and Q axis of every orthogonal frequency is superimposed on the same plot for the noncoherent BFSK and 4FSK modulations. An example of such a signal constellation plot is shown in Figure 41 for an uncoded noncoherent 4FSK system. The time evolution of the sample average bit error probability can also be displayed. Figure 42 shows such a plot for a simulation of an uncoded BPSK system. The bit error plots display the cumulative bit error probability at 100 equally spaced time points during the simulation run. The maximum simulation time must be chosen to allow enough errors to accumulate for a meaningful plot. A simulation time of at least 10,000 is recommended for signal vector modulations. Finally, the input and output signal waveforms of the filters can be displayed. Figure 43 shows such a display of a transmitter filtered signal plus noise waveform at the input to the receiver filter. Each signal waveform plot requires 160 sample points. Therefore the maximum simulation time must be at least 160 to obtain a plot. The graphics subsystem allows only one display to be active at any time. The following signal constellation graphics, bit error probability plotting, and signal waveform display source codes are given in Volume II.

Plotting Function	Source Code File
1) Signal Constellations	PLOTCONS.C
2) Bit Error Probability	PLOTPB.C
3) Transmitter Filter Input	PLTTXINP.C
4) Transmitter Filter Output	PLTTXOUT.C
5) Receiver Filter Input	PLTRCINP.C
6) Receiver Filter Output	PLTRCOUT.C

Page 27 not included

7. CONCLUSIONS

The feasibility of the approach taken in this Phase I effort to develop a visual communications link and network simulation system should be evaluated based on the following considerations:

1) The effectiveness of the user interface in terms of user-friendliness and the quality of the visual displays.

2) The capability of the basic simulation architecture to support a model simulation library that is sufficiently rich enough to be useful for a wide variety of applications.

3) Adequate processing power of the Definicon DSI-750 co-processor enhanced IBM PC/XT compatible computer hardware platform to execute the simulation runs.

We believe that feasibility has been convincingly demonstrated. First, the "Lotus Symphony-like" slash-bar menu interface developed in this effort provides an excellent user-friendly environment for system configuration, controlling the simulation run and accessing the the simulation results in either numerical or graphical form. The flexibility of allowing either mouse-driven or keyboard inputs caters to all possible user preferences in computer usage. The interface and the simulation software will not crash even under a fair amount of user mistakes. For example, it is not possible for the user to specify an unsupported system configuration and crash the software. Many useful facilities are also provided, such as the capability of storing and retrieving system configuration files and the capability of exiting from the menu to the MS-DOS operating system and returning to the menu.

The visual approach is fully supported. Graphical displays of all significant simulation results are provided. The bit error probability displays in the link environment and the average delay and throughput displays in the network environment allow the user to see the dynamic system behavior during a simulation run. This allows for extensive experimentation for system design and also provides valuable insight into system performance evaluations. The signal constellation and signal waveform displays in the link environment give an extra dimension of understanding of the input parameters. For example, the amount of system noise can be seen visually in comparing coded versus uncoded system performance for a given E_b/N_o. Filter waveform displays allow the user to see the transient time domain effects for different design parameters and different filter designs.

The current graphics subsystem implementing the simulation output graphics displays operates under the inherent limitations of the slow IBM PC/XT display hardware. As a result, extensive animated graphics during the simulation run suffers a tremendous increase in execution time. A solution to this problem is to

incorporate a dedicated high-speed graphics processor to implement the graphics screen displays directly. This will allow animated graphics during the simulation run without increasing execution time. Higher resolution color graphics can also be supported in this approach. A Phase II effort can be taken to develop such a graphics subsystem.

The simulation architecture developed in this effort is potentially capable of supporting the development and management of a very large and extensive model library. In this short six month Phase I effort, we have only developed a relatively small model library to demonstrate the capability of the architecture. In the communication link simulation environment, only a fixed topology system consisting of one source transmitting over one channel to one destination was supported. The underlying BLOSIM based simulation operating system is capable, however, of supporting a large variety of other useful topologies. Essentially any communications link topology that can be modelled as an inter-connection of discrete-time functional block operations is supported. For example, co-channel interferences can be modelled by feeding the output of parallel source-encoder-modulator subsystems into the additive channel. Multiple-hop links can also be modelled. The ease with which a large variety of topologies can be supported lies in the flexibility in the BLOSIM environment of specifying the system topology through the UNIVERSE file. The BLOSIM environment also allows a large model library to be developed and maintained easily. These are necessary properties of any useful communication system simulation software system. In this short six month Phase I effort, the BLOSIM environment has allowed the development of 45 functional blocks supporting 175 system configurations. A longer Phase II effort can result in a much larger library of functional blocks and system configurations supporting a variety of topologies in the communications link simulation environment. The Phase II effort should also include the development of software to implement graphical specification of the system configuration by the user.

A general packet communications network has a hierarchial structure consisting of a backbone store-and-forward network with local multiple-access networks at the network entry nodes. In this Phase I effort, we have developed separate software modules for simulating star, uni-directional loop and bi-directional loop store-and-forward networks and for simulating ALOHA, CSMA and Tree collision resolution protocol multiple-access networks. The store-and-forward modules are actually special cases of a general software shell capable of simulating store-and-forward networks of arbitrary topology and size. This was discussed in Section 5 above. The network simulation architecture therefore has the potential capability of handling general packet communications networks. A Phase II effort can be undertaken to implement the capability of simulating any store-and-forward packet communication network. Graphical specification of the network topology by the user would also be desirable here. Finally the ALOHA, CSMA and Tree collision resolution algorithm protocols can be integrated with the general store-and-forward software shell to

model general packet communications networks. Such configurations are useful for studying packet-overlayed architectures for the Army MSE network.

Extensive software testing in this Phase I effort leads us to conclude that the processing power of the supplied 12.5 MHz Definicon DSI-750 co-processor is more than adequate for executing the simulation runs. For example, a 10,000 sample (bit) simulation of an uncoded modulation communications system has an execution time of approximately 40 seconds. Including a (1/2,7) convolutional code to this system (the Viterbi decoder here performs 64 metric comparisons per decoded bit), increases the simulation time to approximately one minute. The slowest execution time occurs in the filtered time-domain BPSK system. The 20X oversampling here requires the filters to generate 40 samples per bit. A 10,000 sample (bit) simulation requires approximately 770 seconds, or roughly 13 minutes. In the network environment, the execution time depends on the number of packets processed during the simulation run. The DSI-750 is capable of processing 4,000 packets per minute.

In conclusion, this SBIR Phase I effort demonstrates the feasibility of implementing a powerful visual communications link and network simulation system. Any IBM PC/XT compatible computer can be upgraded to run this simulation software with the addition of a Definicon 68020 co-processor board. The hardware can be further upgraded on an incremental basis. For example, t h e already extremely fast execution speed can be increased further by increasing the DSI-750 clock rate. The fastest Definicon 68020 co-processor board currently available is a 25 MHz version, which is twice as fast as the 12.5 MHz version. All the current Definicon 68020 boards use the 68882 math co-processor instead of the 68881, providing roughly another 50 percent increase in number crunching power. Existing 68881 boards can be upgraded to the 68882. The performance of our simulation system can be increased substantially with these hardware upgrades.

REFERENCES

1. D. G. Messerschmidt, "A Tool for Structured Functional Simulation," IEEE Journal on Selected Areas in Communications, SAC-2, pp. 137-147, 1984.

2. W.C.Y. Lee, "A new propagation path-loss prediction model for military mobile access," 1985 IEEE MILCOM, pp. 19.2.1-19.2.10

APPENDIX A

A.1 Visual Simulation Software Distribution Disks

The LINKNET Visual Simulation Software is distributed on nine 5 1/4" DSDD floppy diskettes. Disks #1 - #6 contain the operational software while Disks #7 - #9 contain the source code. The diskette contents are as follows:

1) Disk #1 - This disk contains all the link model library object code modules (which have the same file names as the corresponding source code files given in Volume II but with file extensions .OBJ); the run-time BLOSIM object modules - BB.OBJ (BLOCK), BF.OBJ (FIFO), BS.OBJ (SEQUENCER); and C header files (.H file extension) for compiling the UNIVERSE.

2) Disk #2 - This disk contains all the executable link graphics code modules (which have the same file names as the corresponding source code files given in Volume II but with file extensions .E20); and font files (.FNT file extension) for the graphics displays.

3) Disk #3,4 - These two disks contain all the executable network model library code modules and executable graphics code modules. These code module files all have the same file names as the corresponding source code files given in Volume II but with file extensions .E20.

4) Disk #5 - This disk contains the executable interface code LN.EXE and a data base STAR_BAS.DAT used for generating the UNIVERSE in link simulations.

5) Disk #6 - This disk contains sample configurations that can be retrieved. The link configurations have file extensions .LIK and the network configurations have file extensions .NET.

6) Disk #7 - This disk contains the link model library source code listed in Volume II.

7) Disk #8 - This disk contains the network model library source code listed in Volume II. The *.INC files are header files for compiling the *.PAS files.

8) Disk #9 - This disk contains the interface source code listed in Volume III.

A.2 Computer System Configuration and Software Installation

Compaq MSDOS 3.2 must be installed on the 20MB hard disk drive C. It is recommended that directories \DOS, \MOUSE and \LINKNET be created. All the MSDOS 3.2 utilities, the Compaq Deskpro utilities and the Hercules software HGC.COM should be copied in directory \DOS. All the Mouse Systems software should be copied in directory \MOUSE. The root directory should contain the following configuration and automatic batch files.

1) CONFIG.SYS

```
FILES = 22
BUFFERS = 24
DEVICE = ANSI.SYS
DEVICE = C:\DOS\CACHE.EXE 128/BASE
```

2) AUTOEXEC.BAT

```
PATH C:\DOS; C:\MOUSE
KEYBDP
HGC FULL
MSMOUSE /2
```

This ensure that enough file handles are available, sets up a 128 KB disk cache, and initiates the Hercules card and the mouse.

The LINKNET Visual Simulation operational software should be copied in directory \LINKNET. This is accomplished by copying Disks #1 - #6. The following files must also be copied from the Definicon and SVS C disks:

C.E20
CC.E20
JCODE.E20
LINK20.E20
LOAD.COM
CLIB.OBJ
PASLIB.OBJ
All .H files

The visual simulation software is then loaded by entering LN from the keyboard when the current directory is C:\LINKNET.

APPENDIX B

The simulation results that are accessible through the VIEW_DATA option are stored in disk files. These results can be transferred to a printer for hard copy in the MSDOS operating system environment by using either the TYPE or COPY commands. Since these files are overwritten during each simulation run, they must be printed out before the next simulation run is executed. The relevant files are:

1) PBRESULT.DAT - Contains bit and symbol error performance results.

2) TXFILT.DAT - Contains transmitter filter design data.

3) RCFILT.DAT - Contains receiver filter dsign data.

4) POSTTEXT.DAT - Contains delay and throughput results for network configurations.

```
MAIN_MENU
LINK NETWORK DOS EXIT
```

Figure 1

```
MAIN_MENU EXIT
NO YES
```

Figure 2

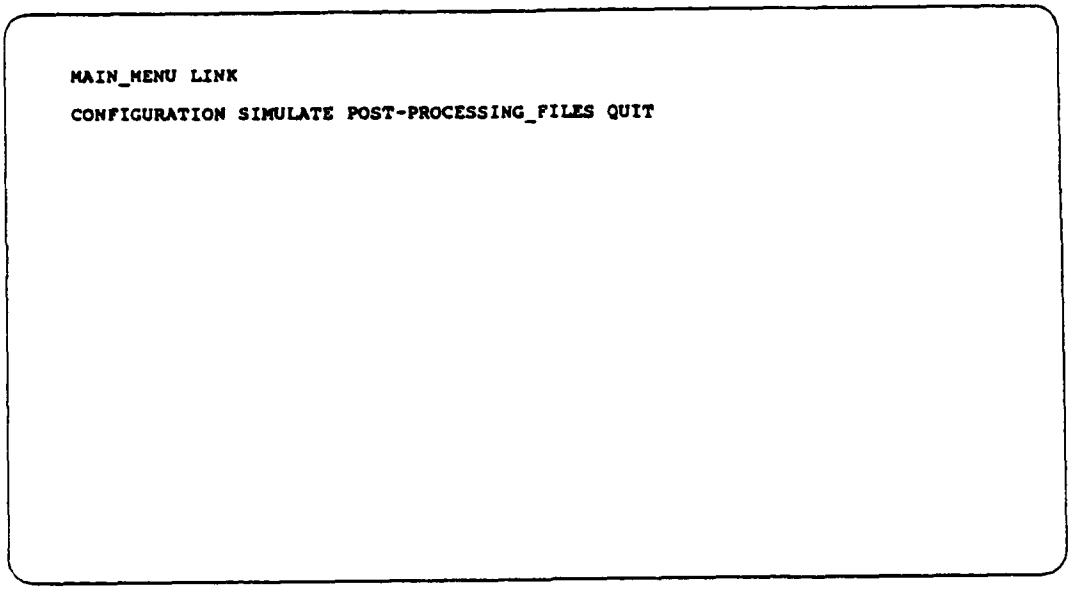

Figure 3

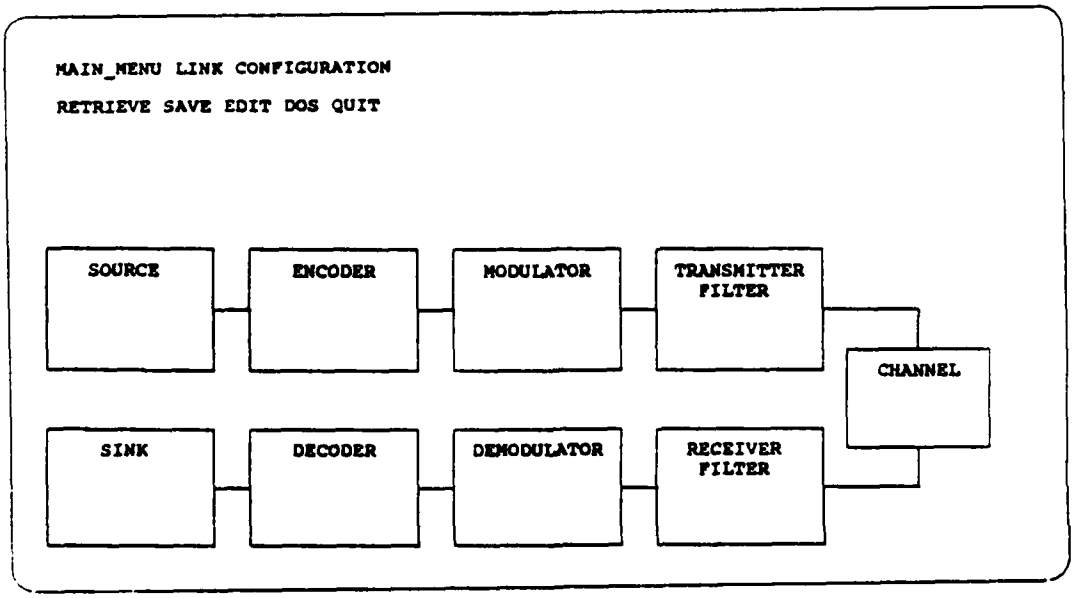

Figure 4

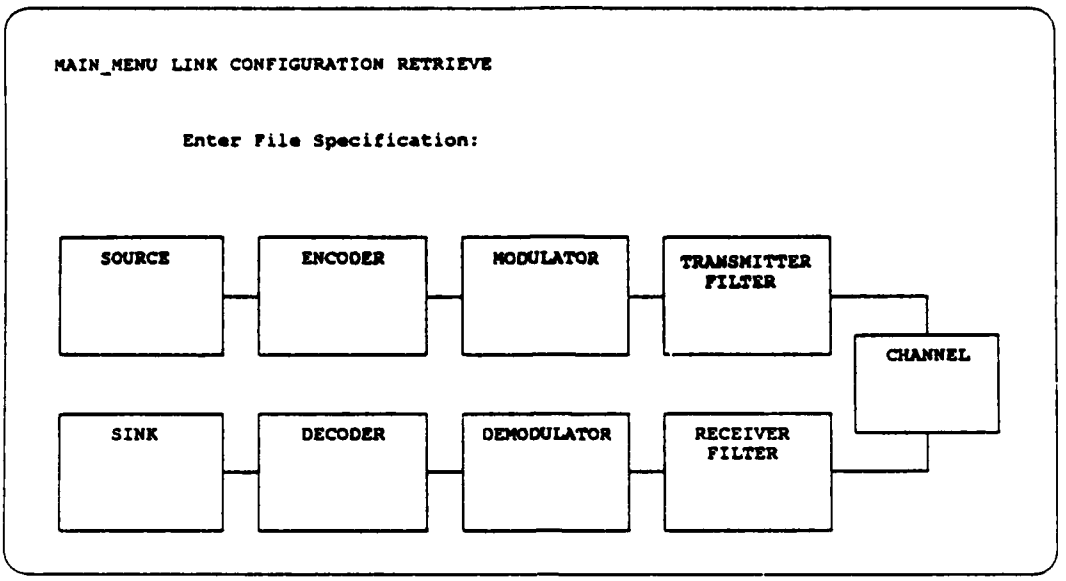

Figure 5a

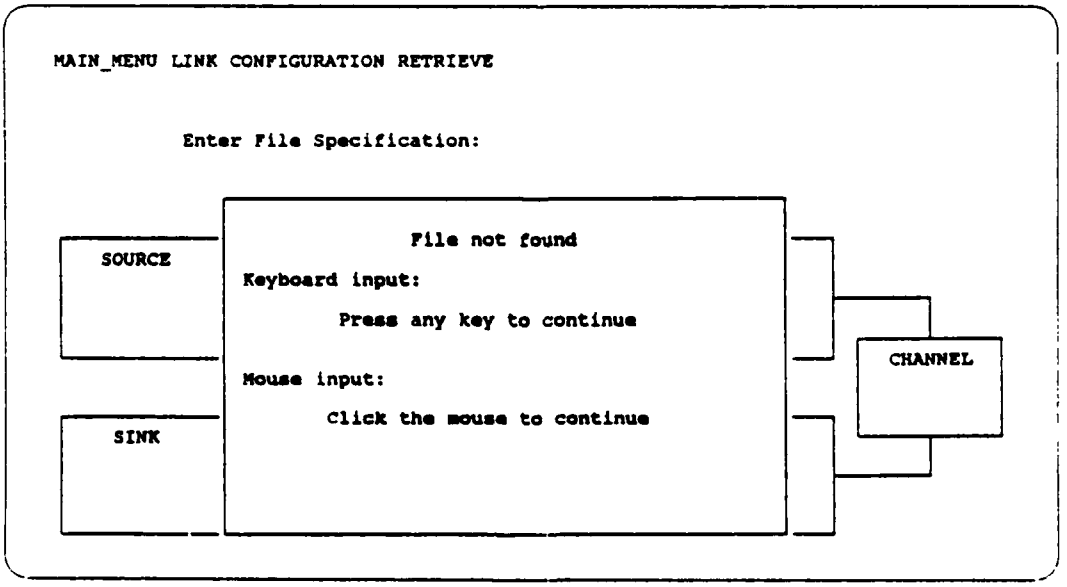

Figure 5b

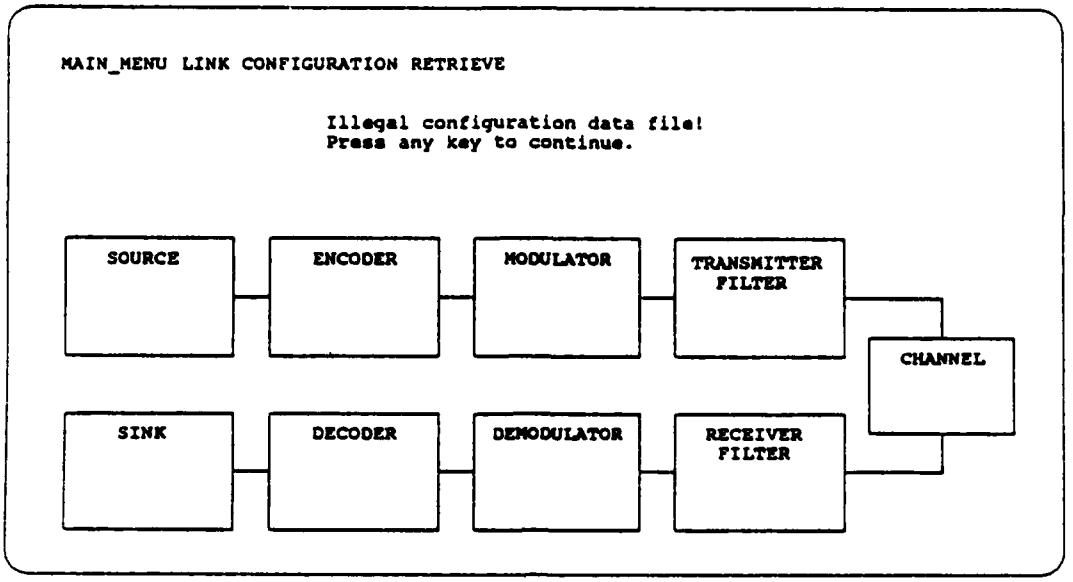

Figure 5c

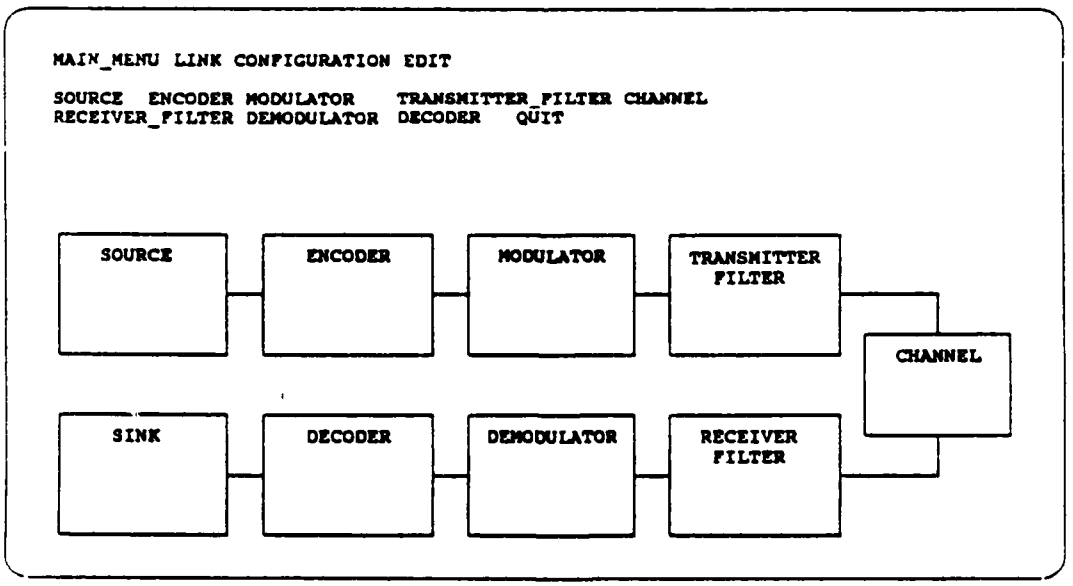

Figure 6

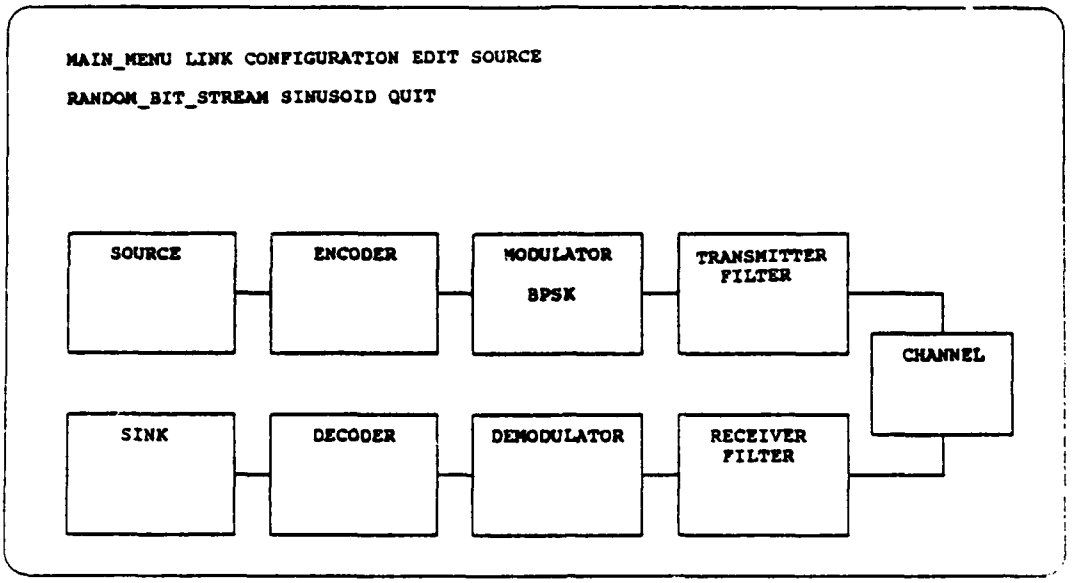

Figure 7

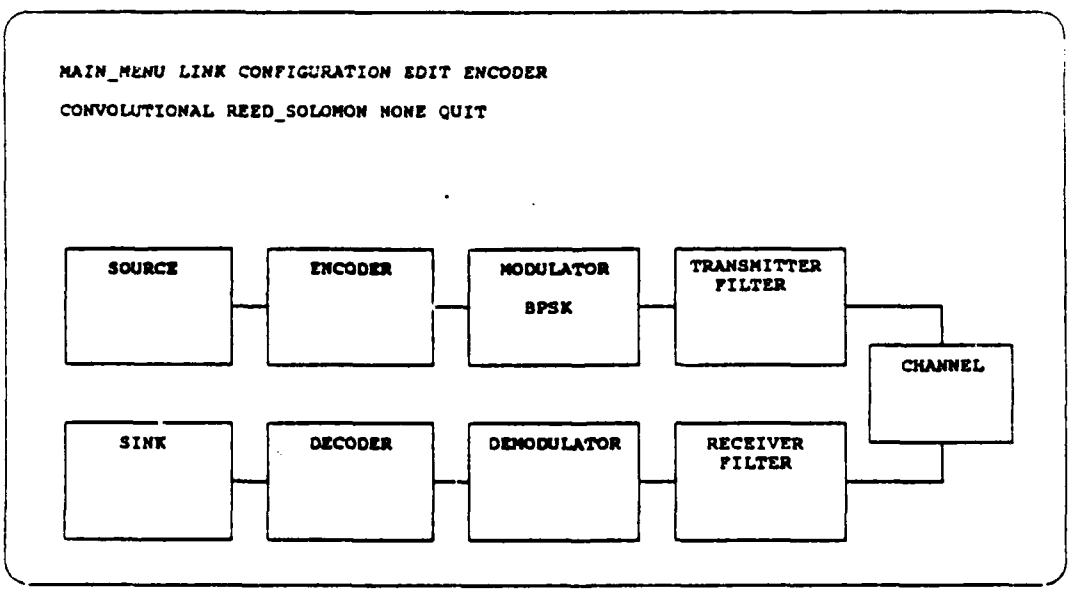

Figure 8

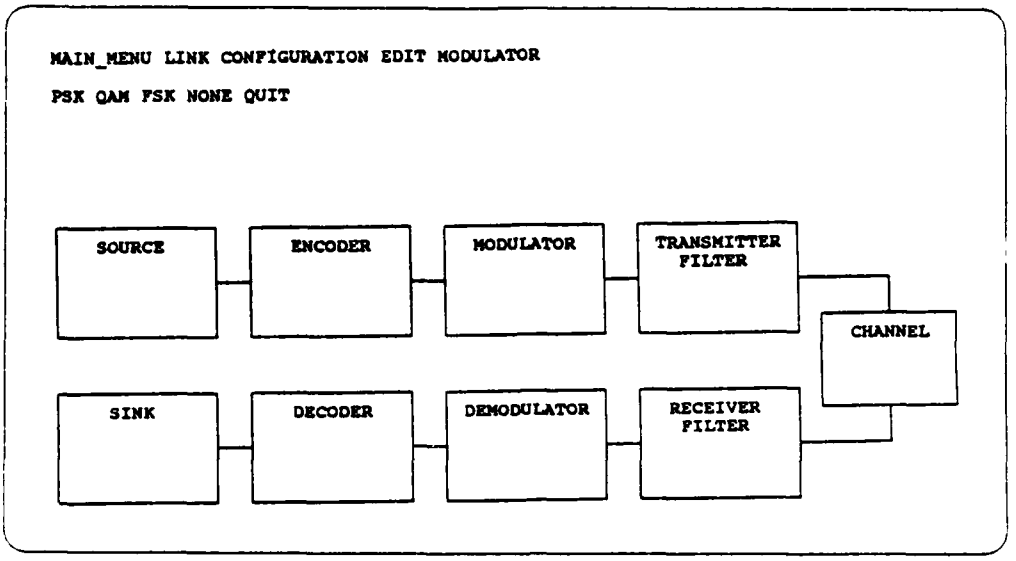

Figure 9

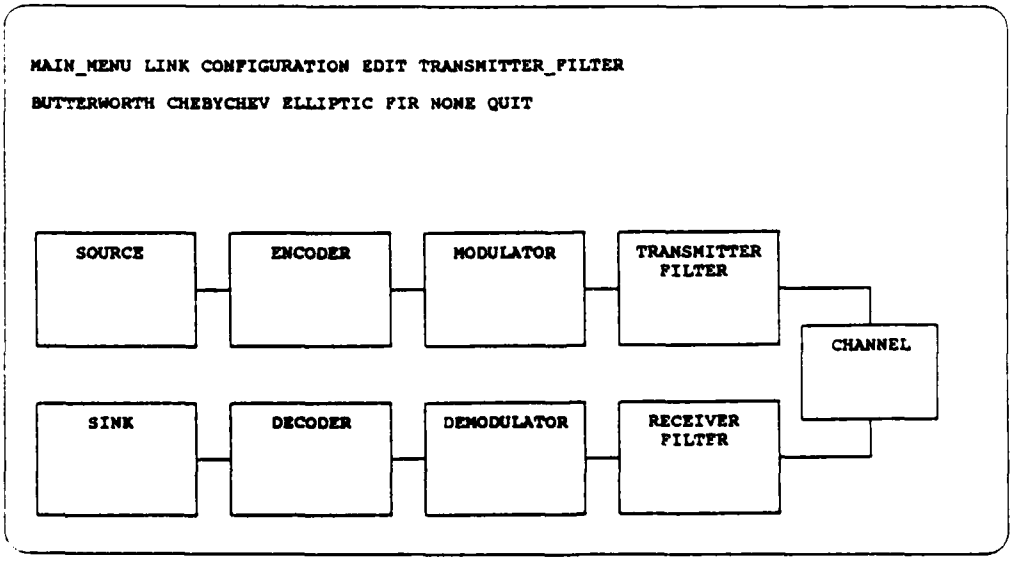

Figure 10

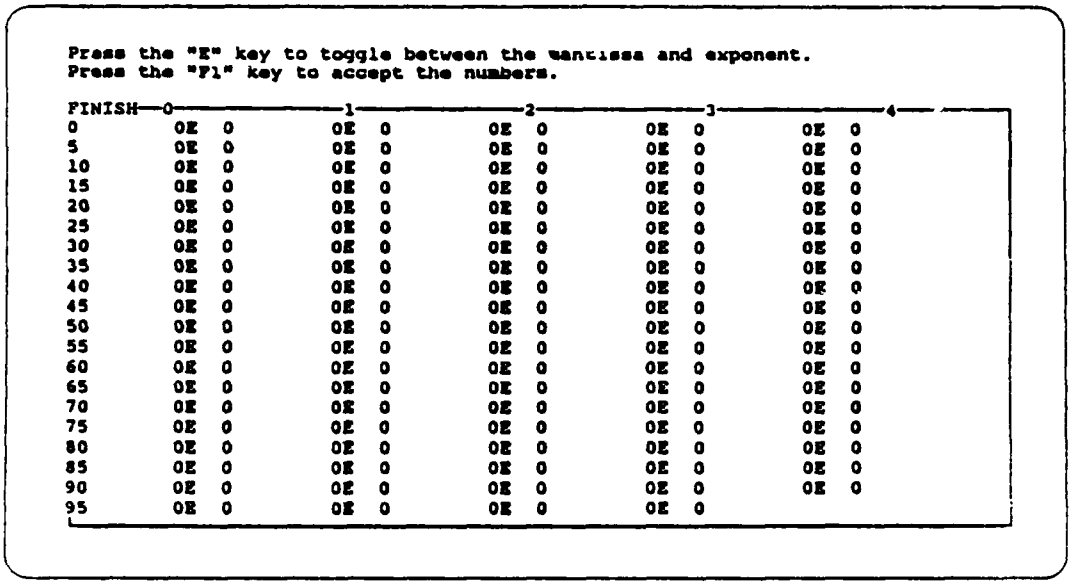

Figure 11

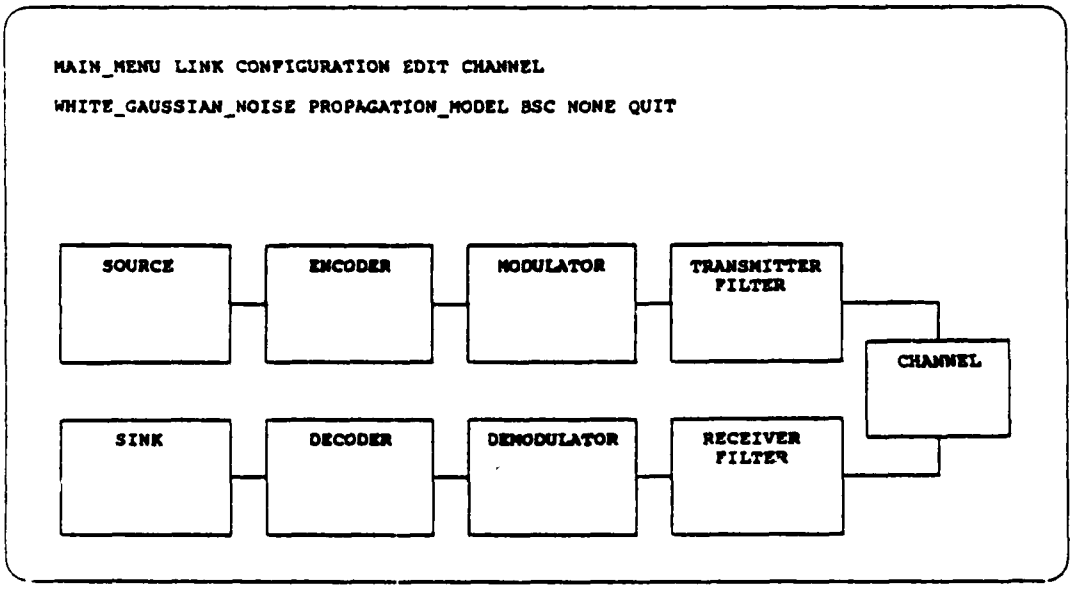

Figure 12

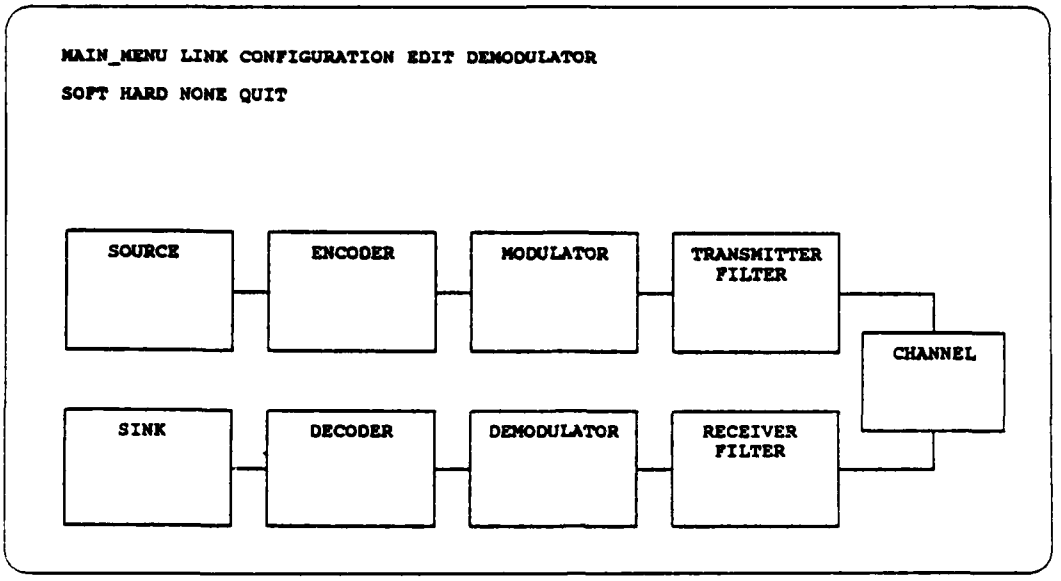

Figure 13

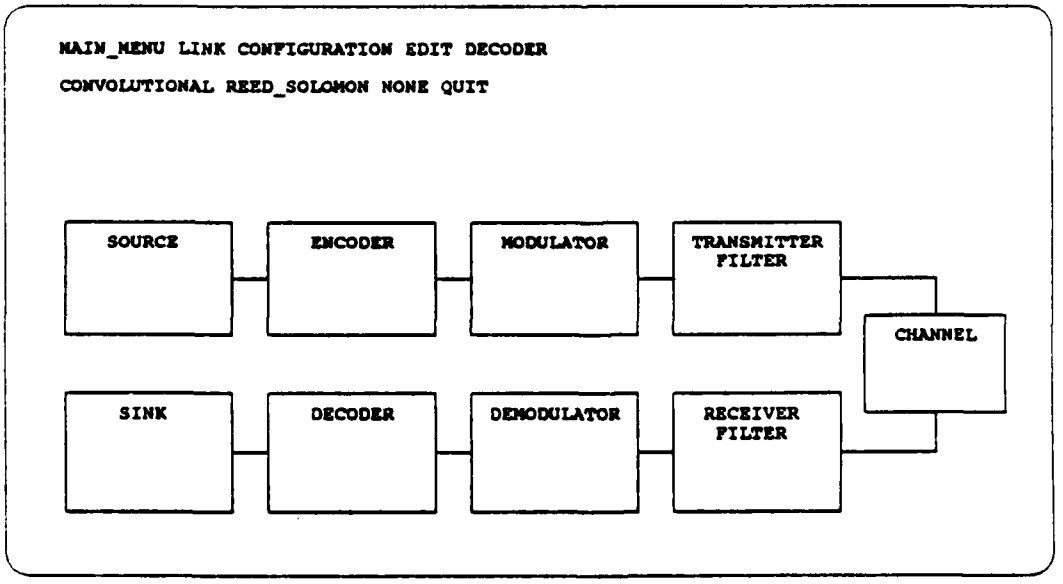

Figure 14

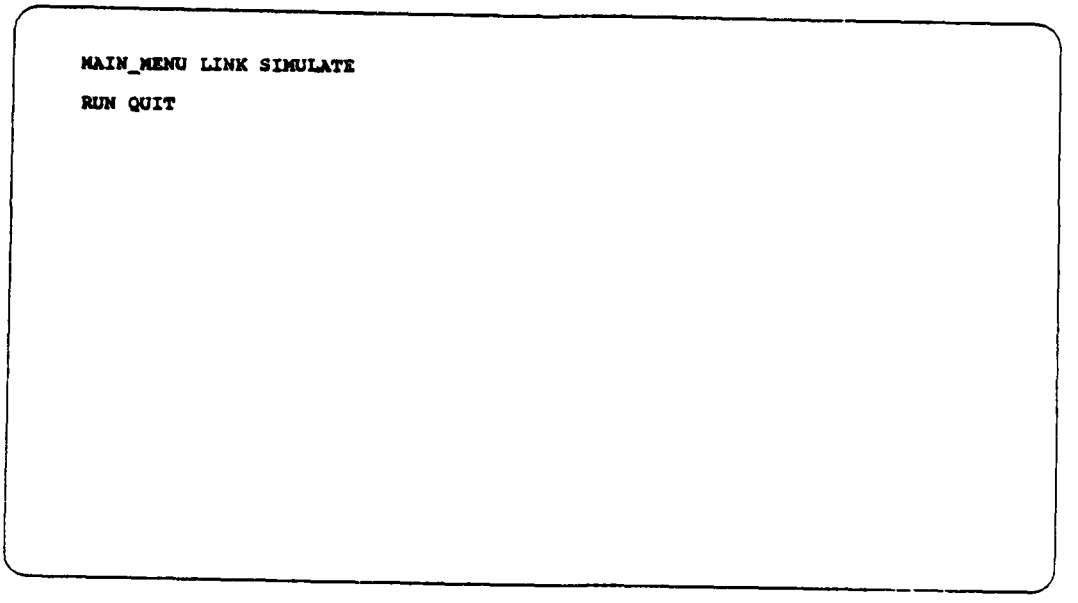

Figure 15a

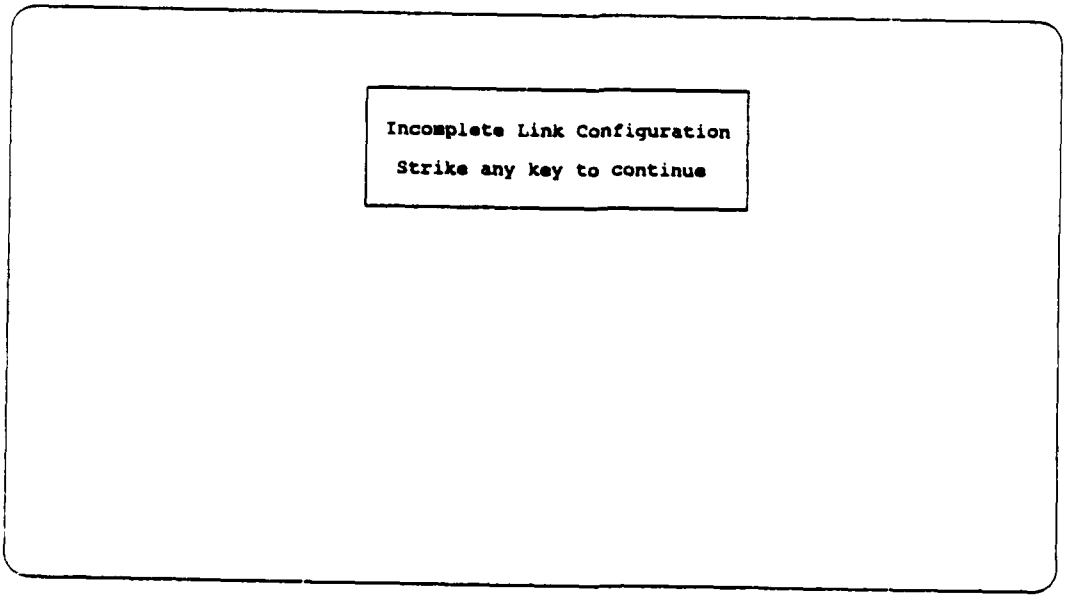

Figure 15b

```
MAIN_MENU LINK POST-PROCESSING_FILES

VIEW_DATA VIEW_GRAPH QUIT
```

Figure 16

```
MAIN_MENU LINK POST-PROCESSING_FILES VIEW_DATA

PERFORMANCE_DATA TRANSMITTER_FILTER_DESIGN_DATA RECEIVER_FILTER_DESIGN_DATA
QUIT
```

Figure 17a

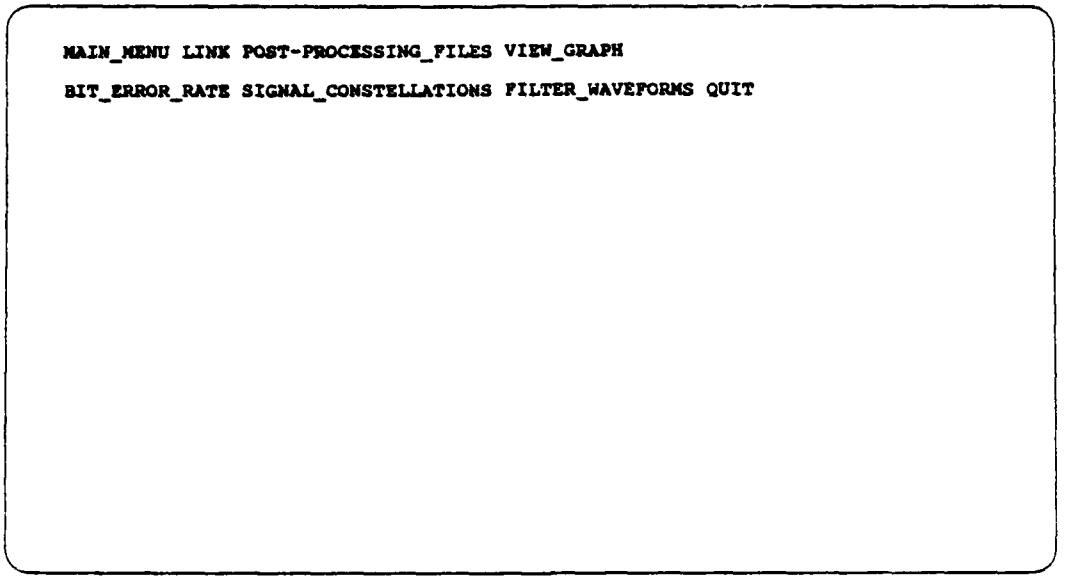

Figure 17b

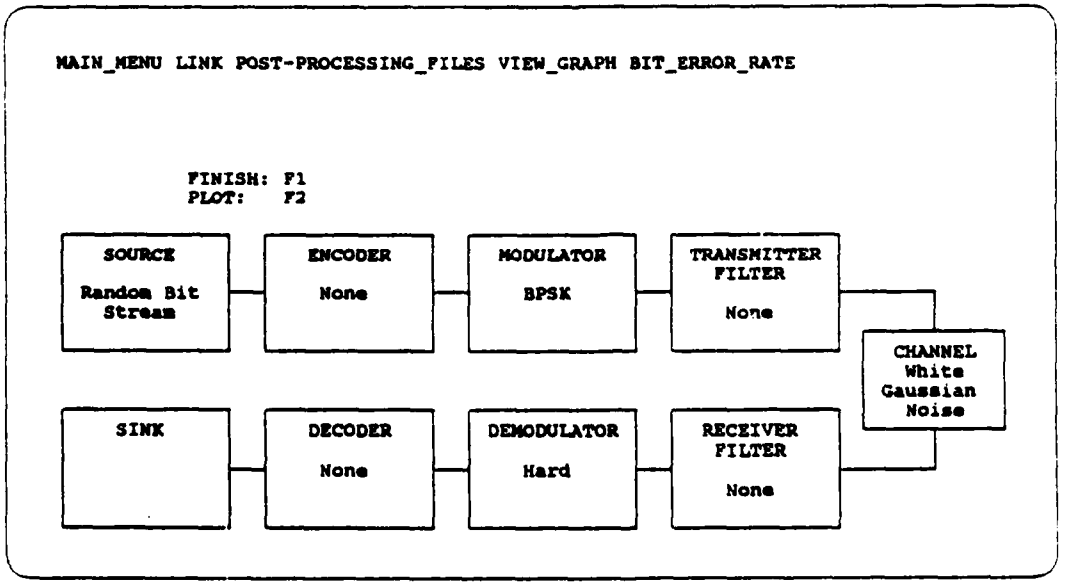

Figure 18

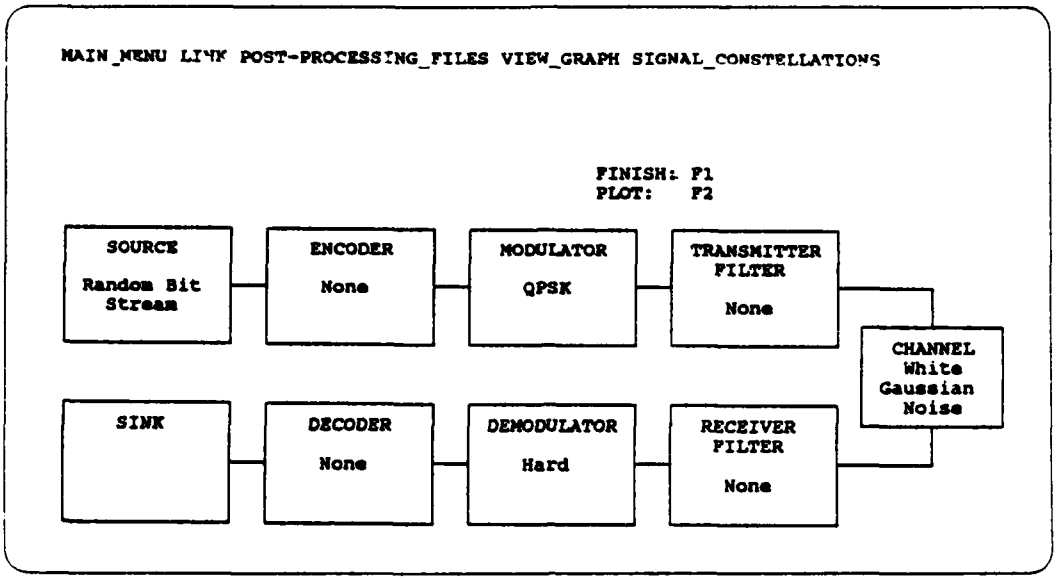

Figure 19

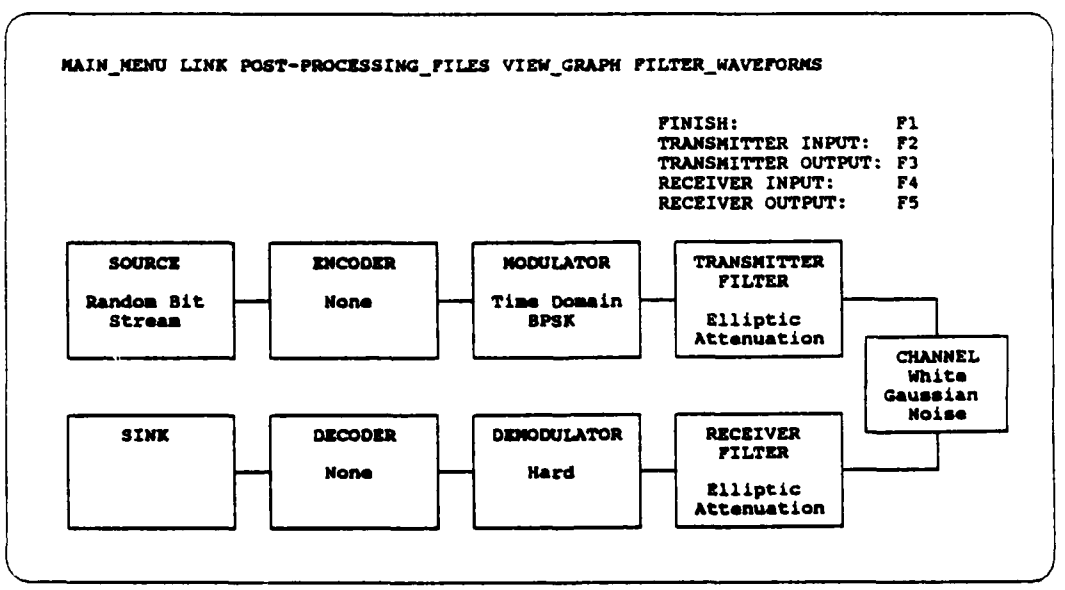

Figure 20

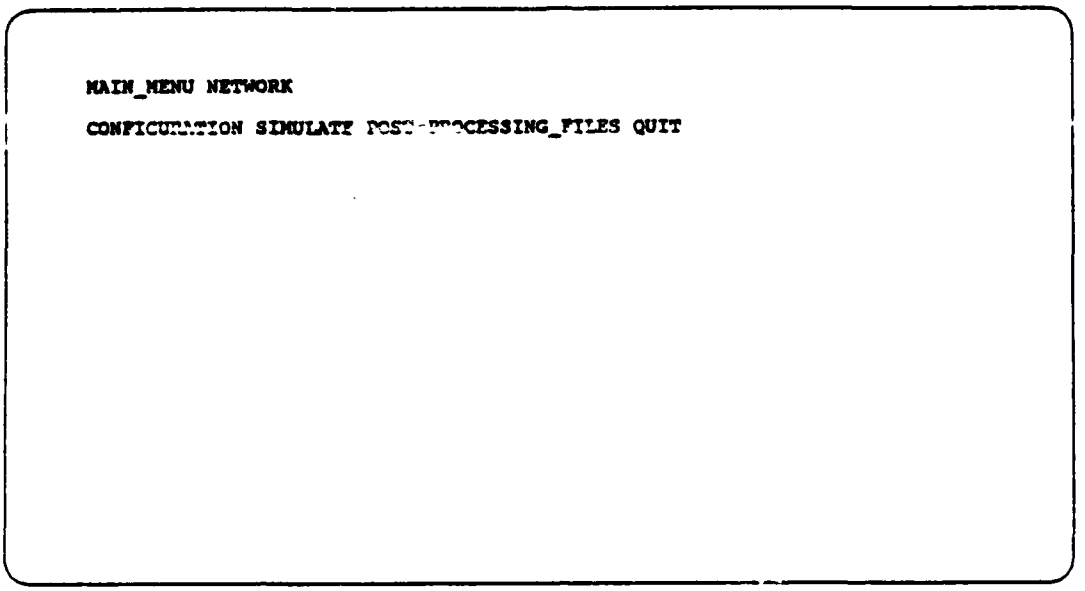

Figure 21

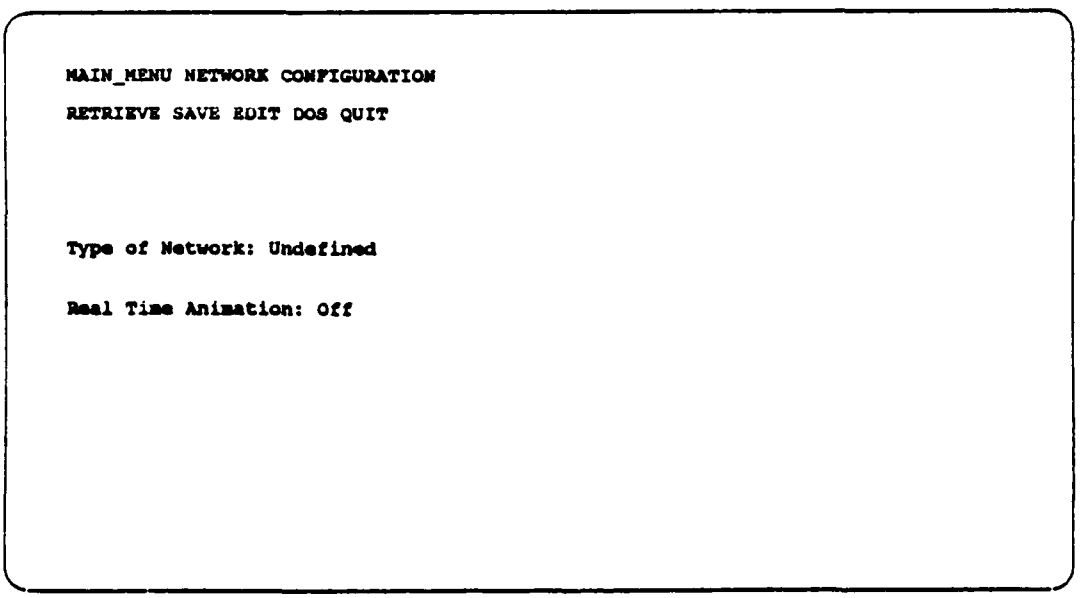

Figure 22

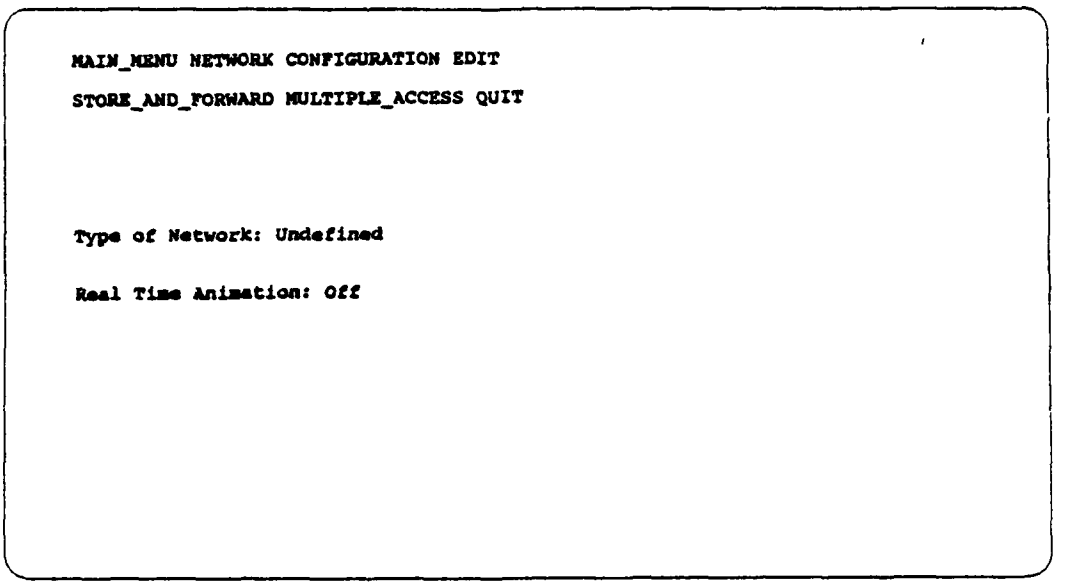

Figure 23

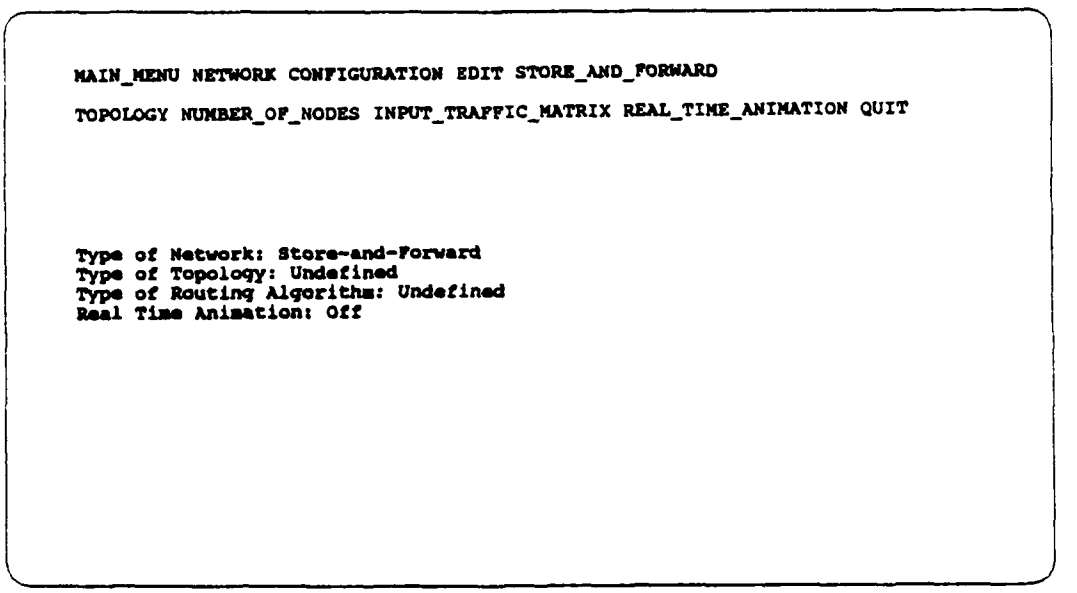

Figure 24

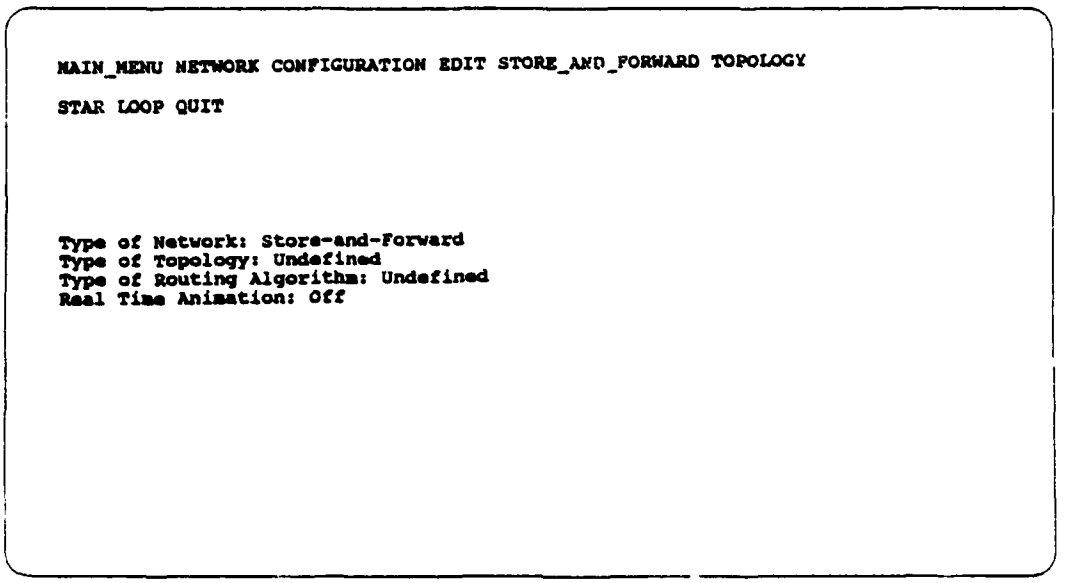

Figure 25

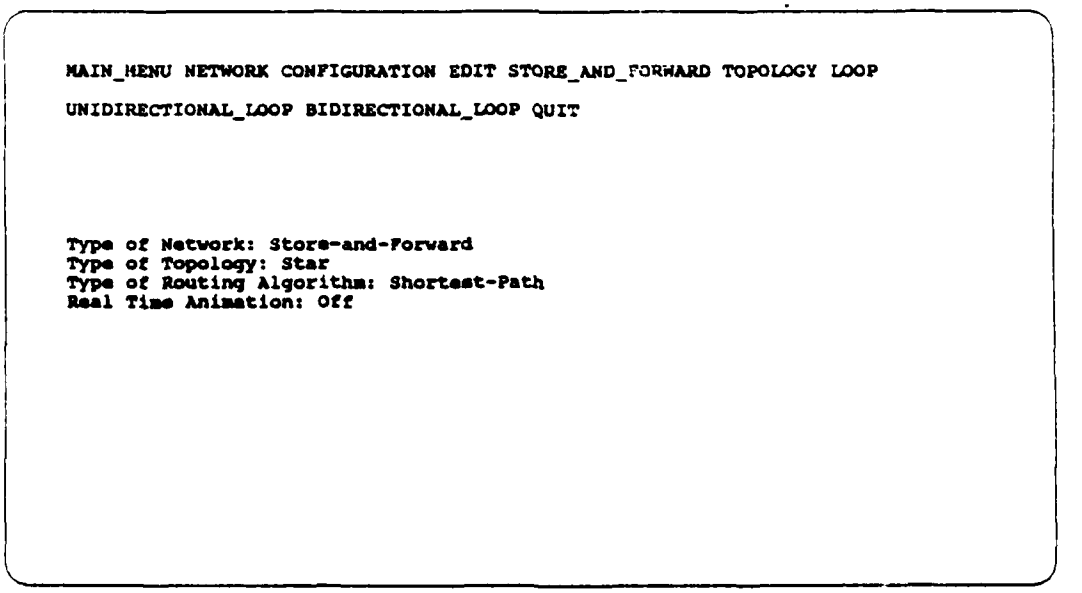

Figure 26a

```
MAIN_MENU NETWORK CONFIGURATION EDIT STORE_AND_FORWARD TOPOLOGY STAR

LINK_CAPACITIES QUIT



Type of Network: Store-and-Forward
Type of Topology: Star
Type of Routing Algorithm: Shortest-Path
Real Time Animation: Off
```

Figure 26b

```
Press the "E" key to toggle between the mantissa and exponent. Press the "-"
key to change the sign of the exponent when the exponent field is active.
Press the "F1" key to accept the numbers.
The unit of the link capacities is packets/sec.

                              FINISH

  ┌──────────┐                    ┌──────────┐
  │2→1  0E 0 │                    │1→2  0E 0 │
  │3→1  0E 0 │                    │1→3  0E 0 │
  │4→1  0E 0 │                    │1→4  0E 0 │
  └──────────┘                    └──────────┘
```

Figure 26c

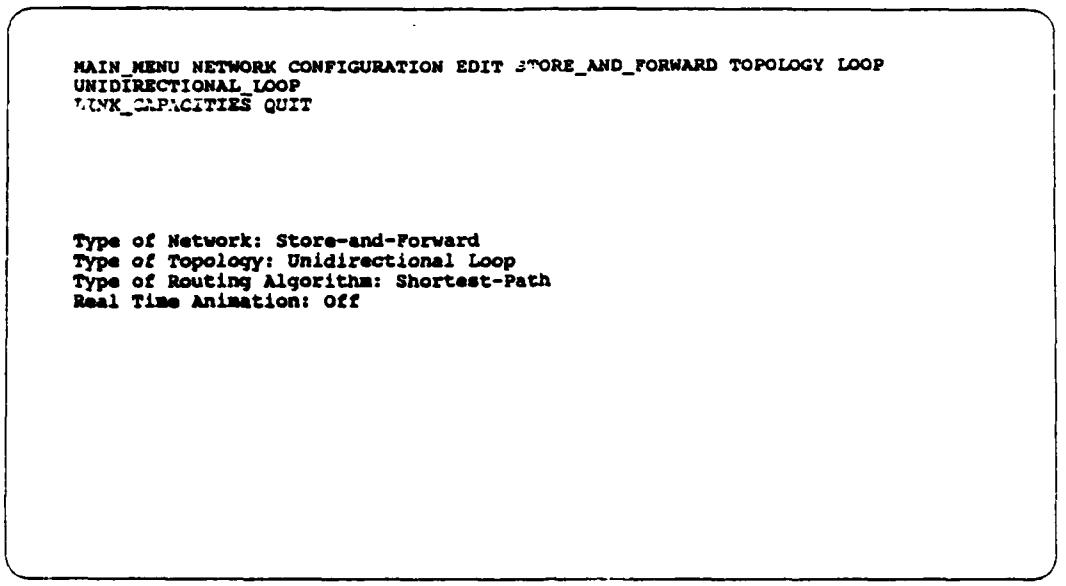

Figure 26d

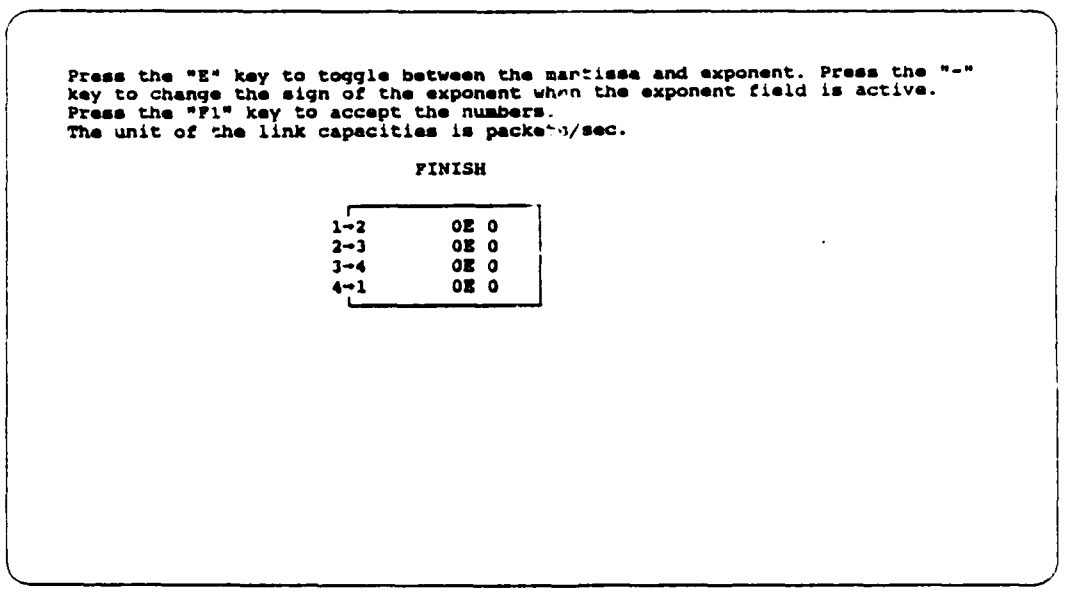

Figure 26e

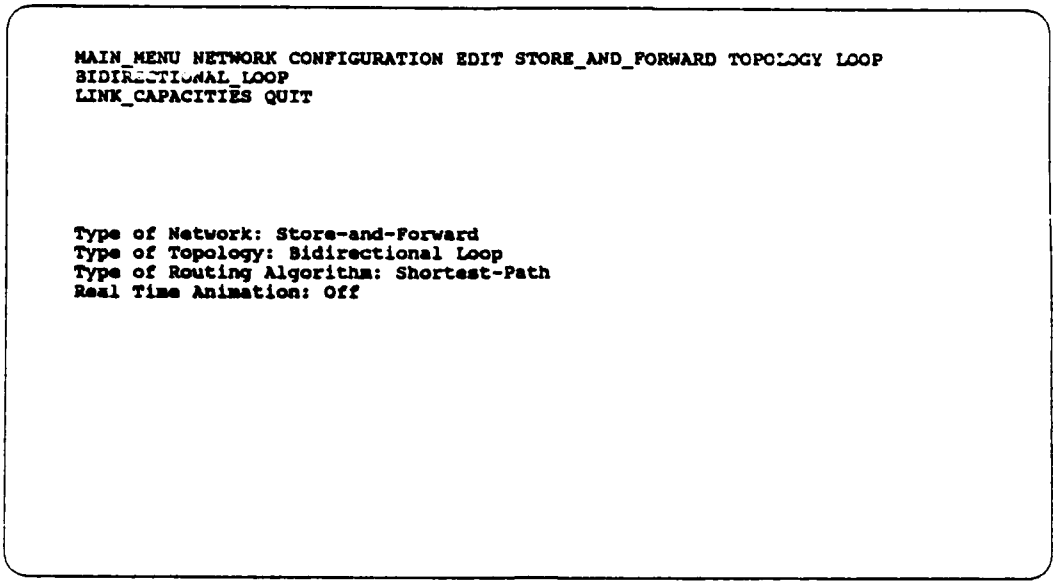

Figure 26f

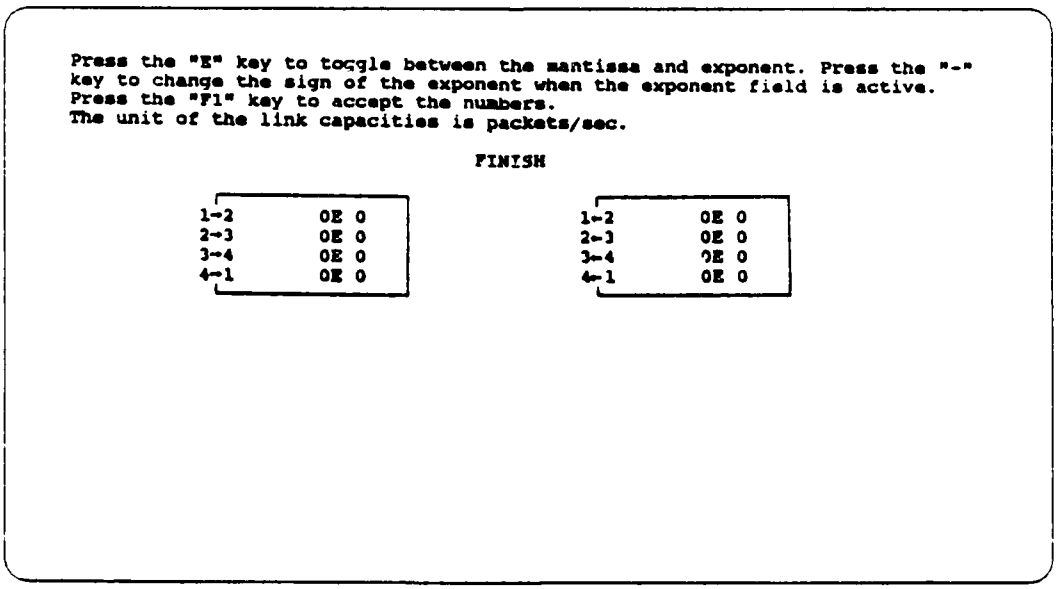

Figure 26g

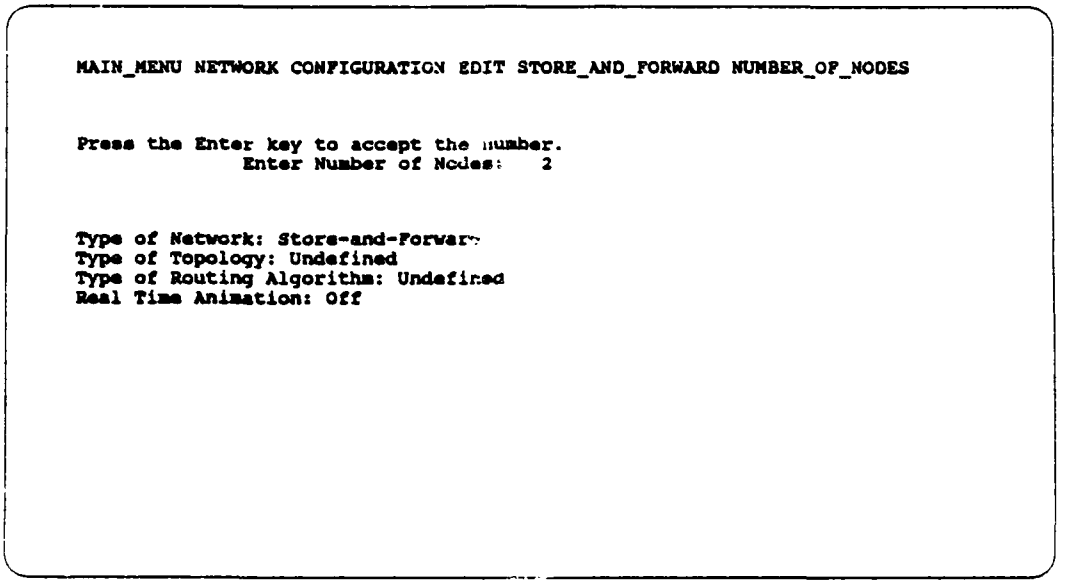

Figure 27

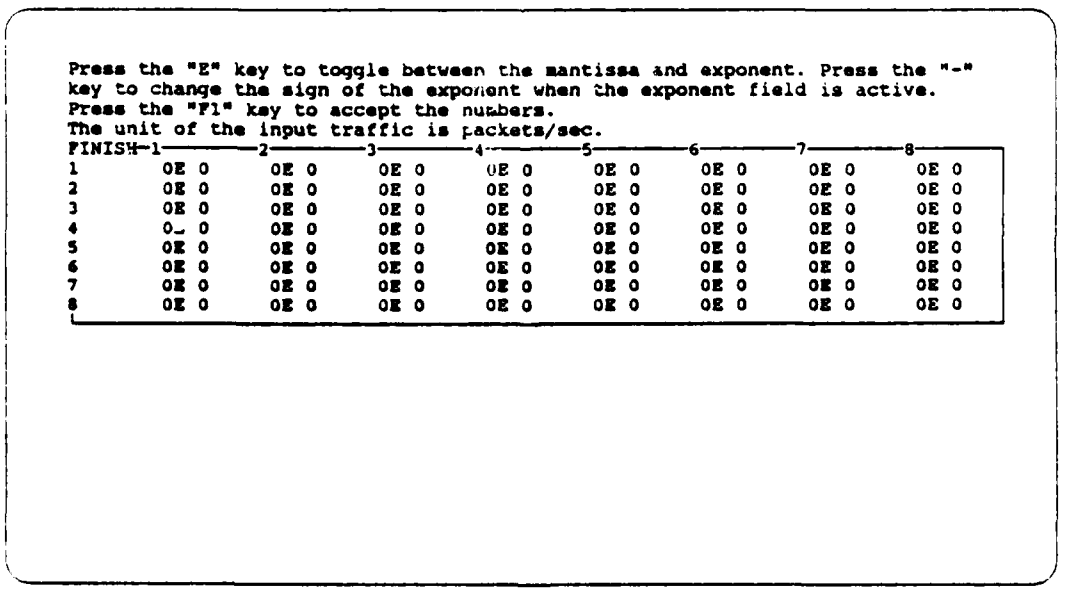

Figure 28

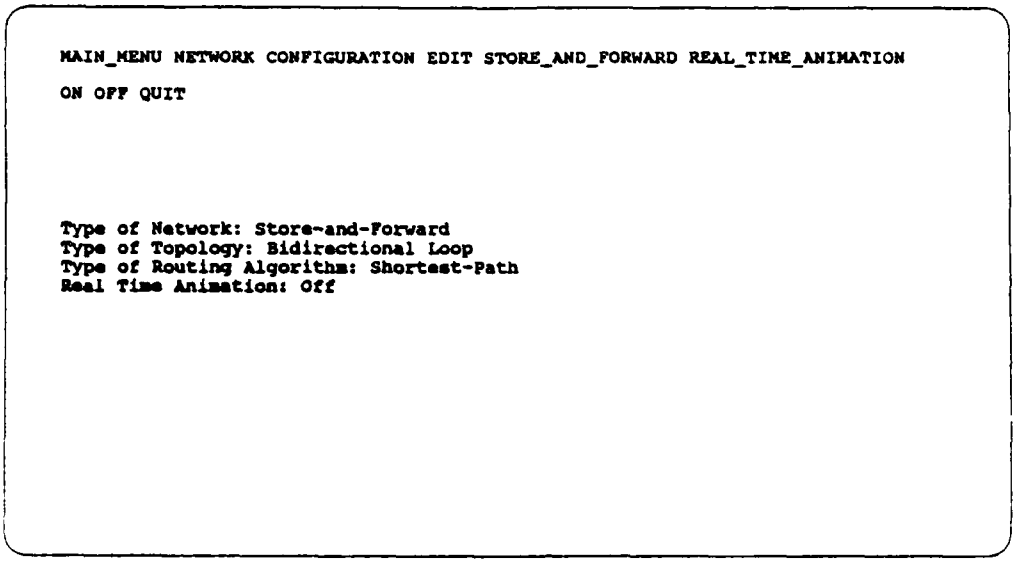

Figure 29

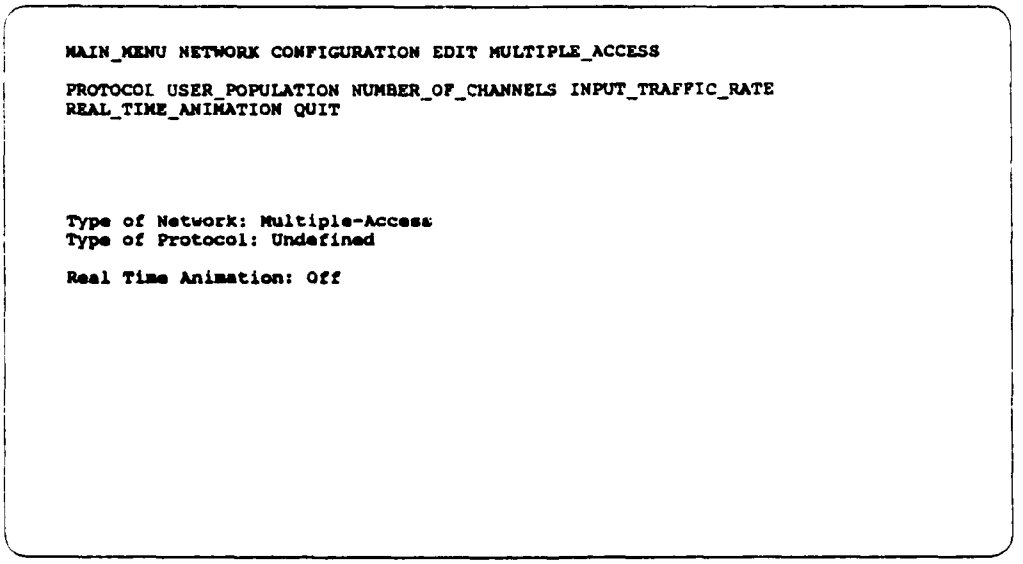

Figure 30

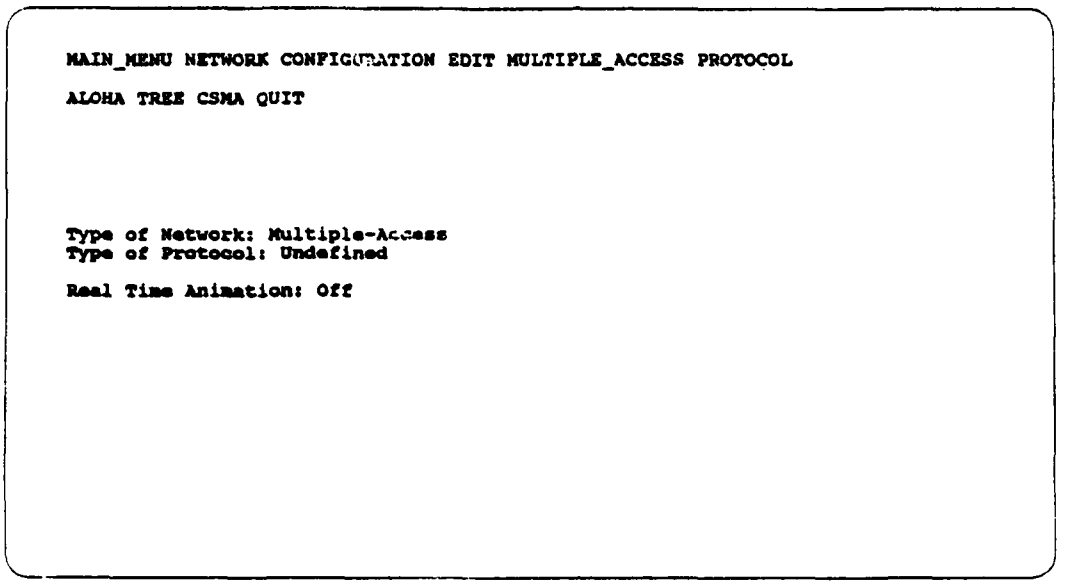

Figure 31a

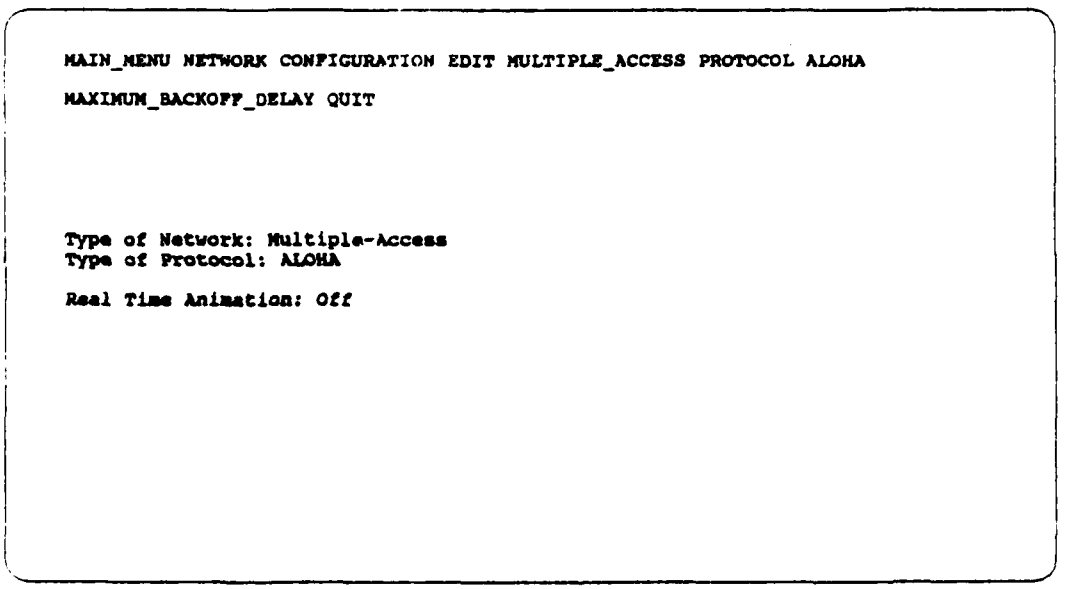

Figure 31b

```
MAIN_MENU NETWORK CONFIGURATION EDIT MULTIPLE_ACCESS PROTOCOL CSMA

MAXIMUM_BACKOFF_DELAY PROPAGATION_RATIO QUIT




Type of Network: Multiple-Access
Type of Protocol: CSMA

Real Time Animation: Off
```

Figure 31c

```
MAIN_MENU NETWORK CONFIGURATION EDIT MULTIPLE_ACCESS USER_POPULATION


Press the Enter key to accept the number.
          Enter Number of Stations:   1


Type of Network: Multiple-Access
Type of Protocol: CSMA

Real Time Animation: Off
```

Figure 32

```
MAIN_MENU NETWORK CONFIGURATION EDIT MULTIPLE_ACCESS NUMBER_OF_CHANNELS


Press the Enter key to accept the number.
          Enter Number of Channels:   1


Type of Network: Multiple-Access
Type of Protocol: ALOHA

Real Time Animation: Off
```

Figure 33

```
MAIN_MENU NETWORK SIMULATE

RUN OUTPUT_DISPLAY QUIT
```

Figure 34

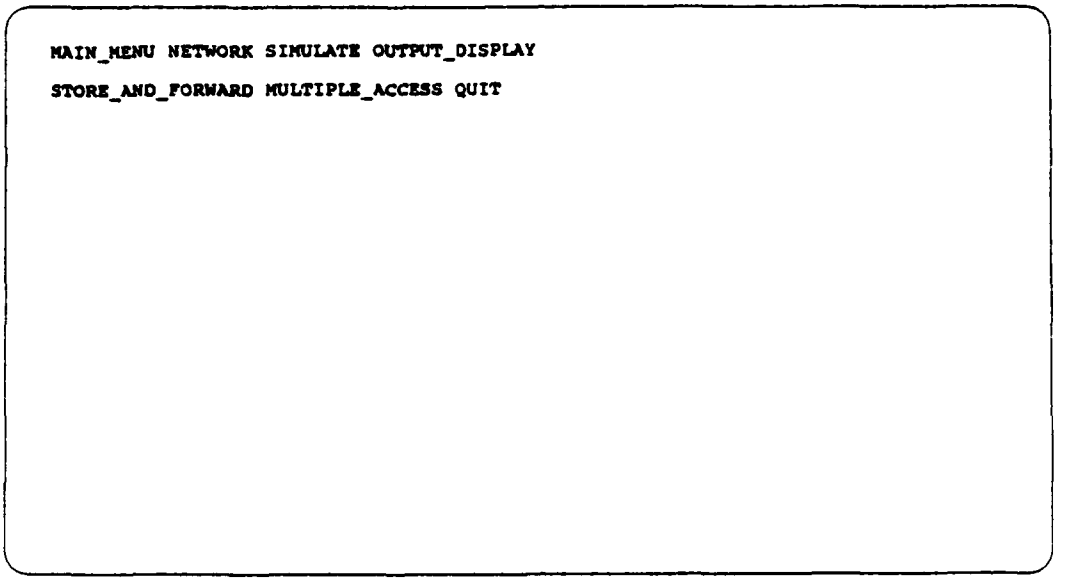

Figure 35

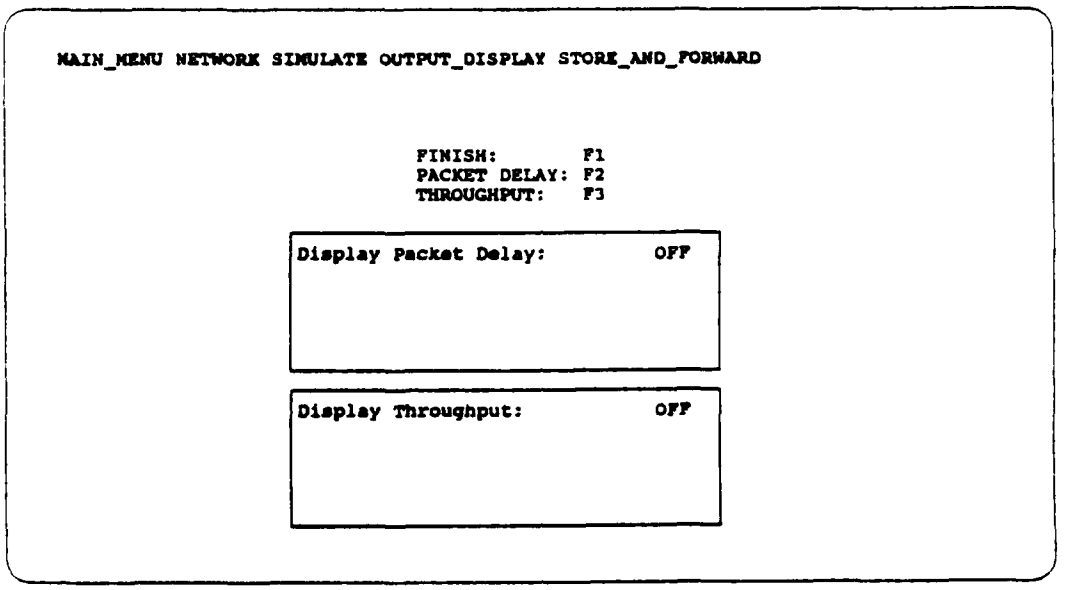

Figure 36

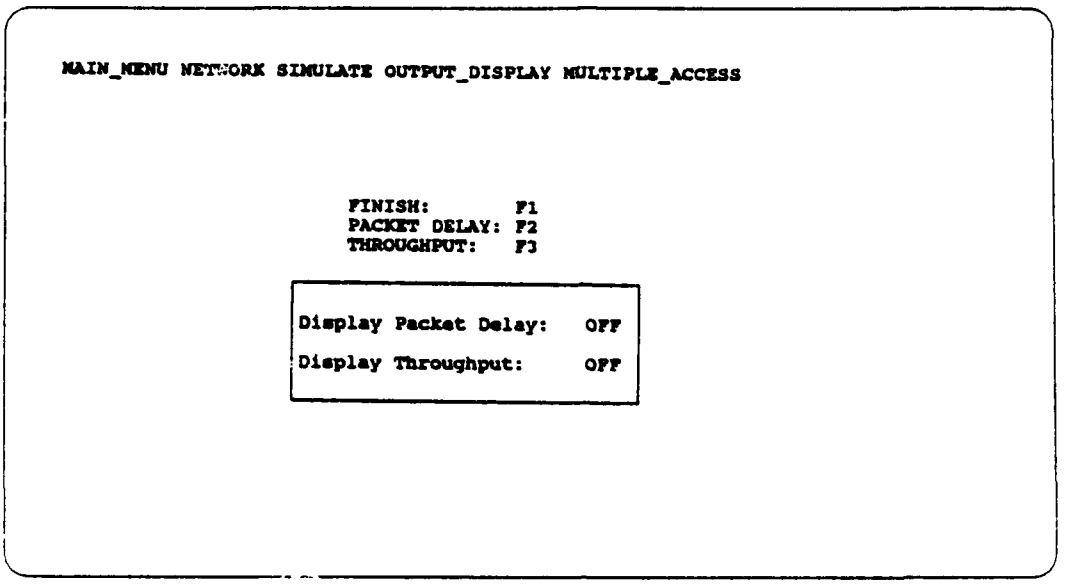

Figure 37

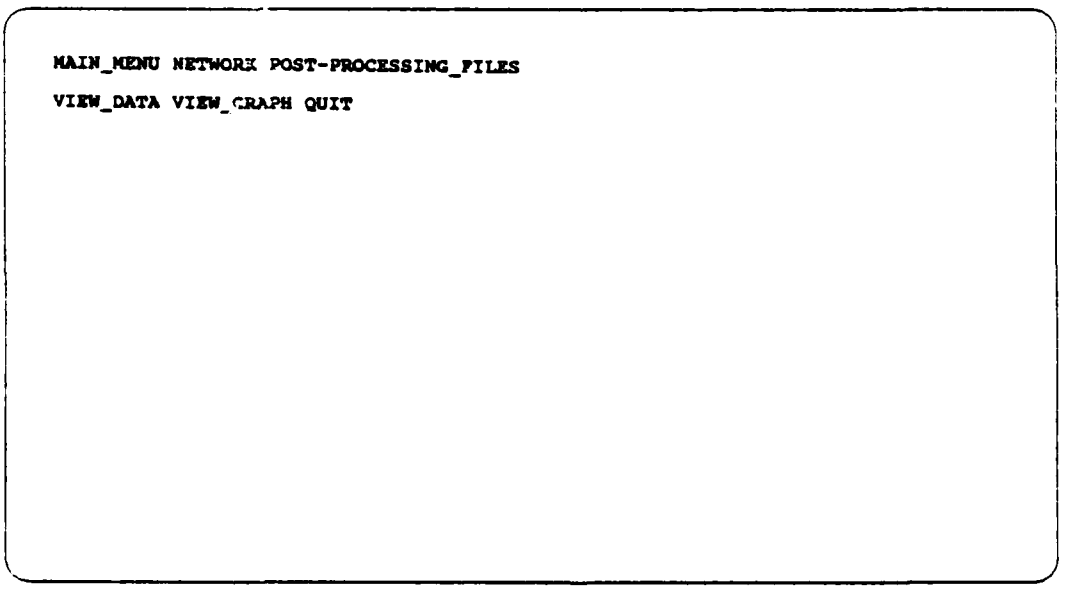

Figure 38

MAIN_MENU NETWORK POST-PROCESSING_FILES VIEW_GRAPH

PACKET_DELAY THROUGHPUT QUIT

Figure 39

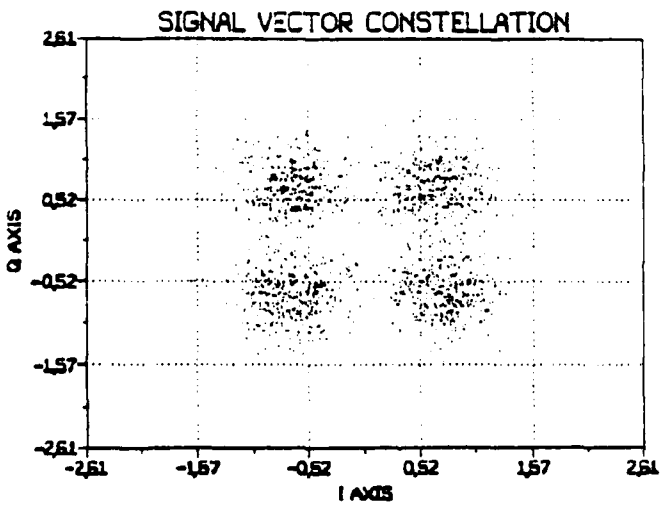

Figure 40

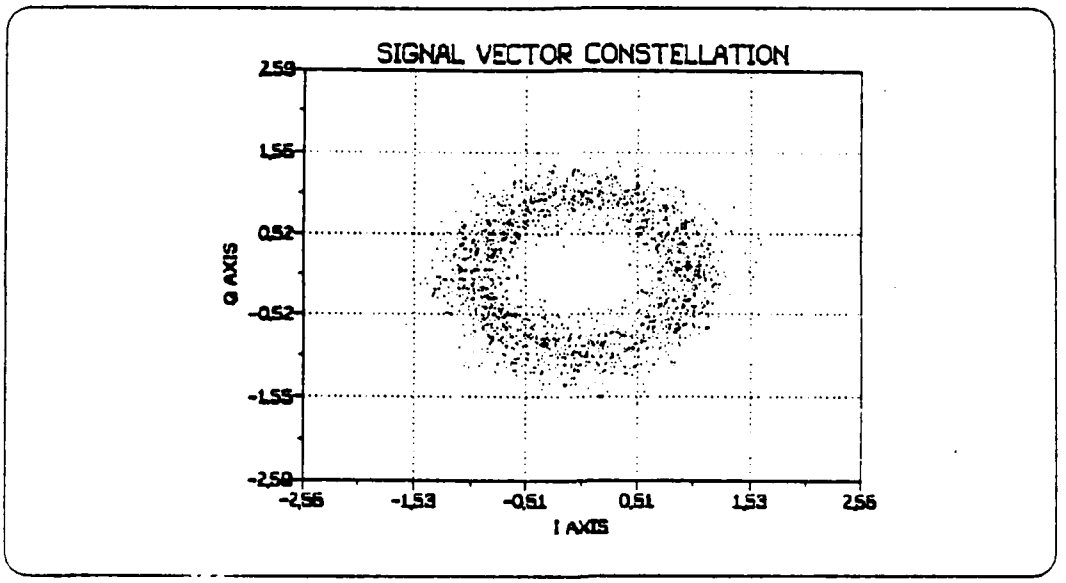

Figure 41

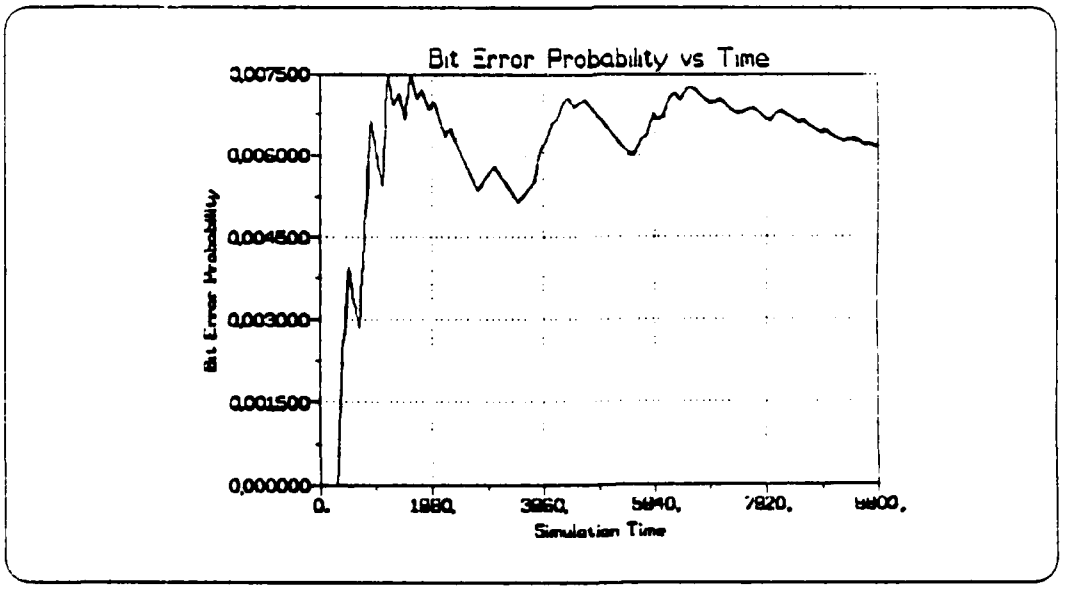

Figure 42

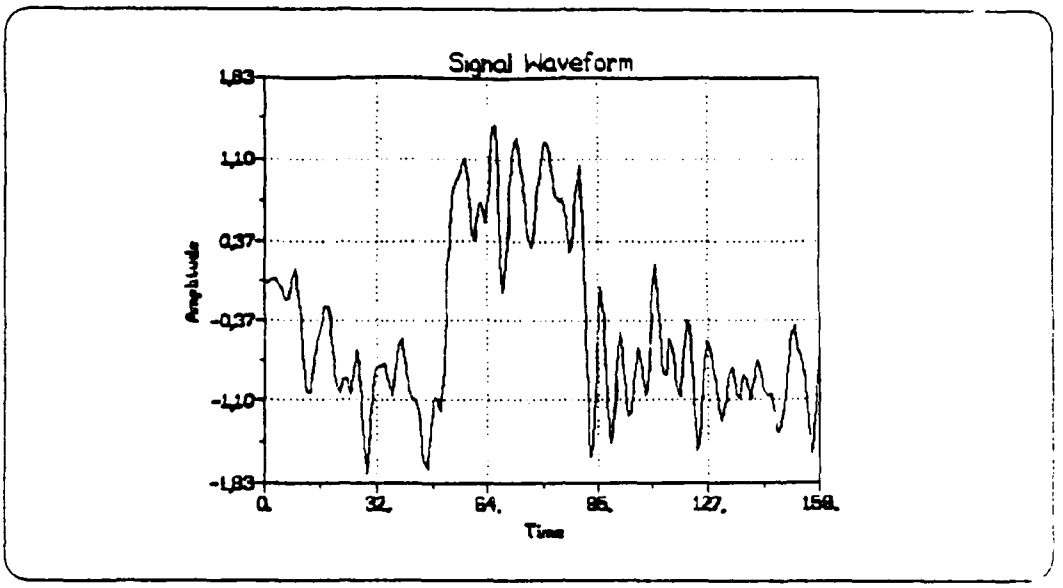

Figure 43

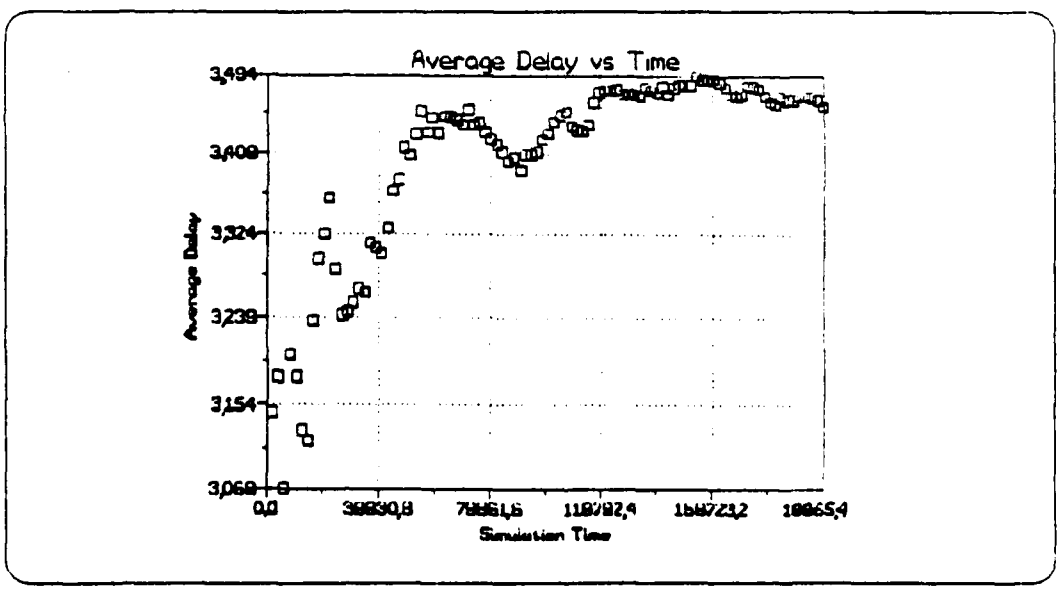

Figure 44

Page 62 not included